STAT40150 Multivariate Analysis Assignment 1

Fanahan McSweeney - 20203868

February 27, 2021

Install / Load Packages

Note that code is included in the chunk below to install all packages used in the file, but these commands have been commented out. Please uncomment commands related to any packages that have not yet been installed.

```
## Install the relevant packages (UNCOMMENT THESE IF PACKAGES ARE NOT INSTALLED!)
# install.packages("knitr")
# install.packages("corrplot")
# install.packages("car")
# install.packages("cluster")
# install.packages("e1071")
# install.packages("MASS")
# install.packages("class")

## Load relevant libraries
library(knitr)
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
library(car)
```

```
## Loading required package: carData
```

```
library(cluster)
library(e1071)
library(MASS)
library(class)
```

Data Preparation

After loading the *Milk_MIR_Traits_data.csv* file, some initial tidy up of the data must be done before performing analysis.

Firstly, a single observation, chosen at random, is dropped from the data set. From inspecting the columns in the input data, there are several recorded variables that are not useful for the purpose of this project, and so these can be dropped from the data before proceeding.

The *str* function can be used to look at the structure of a given data frame and give a quick insight into the data contained in each of its columns. Below, the first 18 columns of the reduced data frame are inspected, which include all protein and technological traits for the data.

```
# Load data from .csv file (in "Data" folder within working directory)
rawdata <- read.csv("Data/Milk_MIR_Traits_data.csv")

# set seed to my student no.
set.seed(20203868)
# Generate random integer between 1 and number of rows in the data set
drop_index <- sample(nrow(rawdata), 1)
# Remove row of the randomly generated index from the data
rawdata <- rawdata[-drop_index, ]

# Remove any columns that will not be used during the analysis etc.
# Save in a new data frame called "milkdata"
milkdata <- rawdata[-c(5:6, 14:39, 41:43, 46, 51)]

# Look at the structure of the data
str(milkdata[1:18])
```

```
## 'data.frame':    430 obs. of  18 variables:
##  $ Breed               : chr  "Hol Fri" "Hol Fri" "JEX-" "FRX-" ...
##  $ Date_of_sampling    : chr  "20/08/2013" "20/08/2013" "20/08/2013" "20/08/2013" ...
##  $ Parity              : int  5 5 5 4 5 2 2 2 1 2 ...
##  $ Milking_Time        : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ kappa_casein        : num  5.31 6.11 7.06 8.29 7.3 ...
##  $ alpha_s2_casein     : num  3.13 3.723 0.657 6.146 4.022 ...
##  $ alpha_s1_casein     : num  12.1 12.6 12.5 21.8 16.8 ...
##  $ beta_casein         : num  9.71 11.08 11.49 17.22 14.35 ...
##  $ alpha_lactalbumin   : num  0.933 0.916 1.074 1.254 0.983 ...
##  $ beta_lactoglobulin_a: num  2.096 0.903 2.116 3.065 2.259 ...
##  $ beta_lactoglobulin_b: num  2.46 5.17 1.36 0 2.99 ...
##  $ Casein_micelle_size : num  159 146 125 243 146 ...
##  $ Heat_stability      : num  31 31 31 9.95 21.6 ...
##  $ pH                  : num  NA 6.66 6.63 6.6 6.55 ...
##  $ RCT                 : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ k20                 : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ a30                 : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ a60                 : num  NA NA NA NA NA NA NA NA NA NA ...
```

There are a number of categorical covariates recorded in the given data set. For this project, the following categorical variables will be considered:

- *Breeds*
- *Parity*
- *Date_of_sampling*
- *Milking_Time*

For each of these variables, there is some level of tidy up required.

Firstly, looking at a frequency table of the *Breeds* data shows 11 different categories recorded. However, some of these categories appear to overlap, for example *hox*, *HOX*, and *HOX-* are all likely representative of a Holstein Cross breed.

Below, I have grouped any of the categories that I judged to be identical, and I have renamed each category appropriately. I have also converted the column type to a factor. A frequency table of the updated *Breed* categories is shown below.

```
# Replace all Breeds containing "FRX" with "FriesianX"
milkdata$Breed[sapply("frx", grepl, milkdata$Breed, ignore.case = TRUE)] <- "FriesianX"
# Replace all Breeds containing "hox" with "HolsteinX"
milkdata$Breed[sapply("hox", grepl, milkdata$Breed, ignore.case = TRUE)] <- "HolsteinX"
# Replace all Breeds containing "jex" with "JerseyX"
milkdata$Breed[sapply("jex", grepl, milkdata$Breed, ignore.case = TRUE)] <- "JerseyX"
# Replace all Breeds equalling "Hol Fri" with "HolsteinFriesian"
milkdata$Breed[milkdata$Breed=="Hol Fri"] <- "HolsteinFriesian"
# Replace all Breeds equalling "je" with "Jersey"
milkdata$Breed[milkdata$Breed=="JE"] <- "Jersey"
# Replace all Breeds equalling "MO" with "Montbelliarde"
milkdata$Breed[milkdata$Breed=="MO"] <- "Montbelliarde"
# Replace all Breeds equalling "NR" with "NorwegianRed"
milkdata$Breed[milkdata$Breed=="NR"] <- "NorwegianRed"
# Replace unfilled entries  for Breeds with "Unknown"
milkdata$Breed[milkdata$Breed==""] <- "Unknown"

# Make Breed column a factor
milkdata$Breed <- as.factor(milkdata$Breed)

# View frequency table for Breed data
kable(t(table(milkdata$Breed)), align = 'c')
```

FriesianX	HolsteinFriesian	HolsteinX	Jersey	JerseyX	Montbelliarde	NorwegianRed	Unknown
18	296	14	60	28	1	12	1

The *Parity* column contains the number of different times that the cow has had offspring for each observation. Values in the data set range from 1 to 11, but the frequency of values greater than 5 is significantly lower than values less than 6.

Therefore, I decided to group *Parity* values greater than 5 into a single category called *6+*, and I have again changed the column's data type to a factor.

```r
# Set all values of Parity greater than 5 to "6+"
milkdata$Parity[milkdata$Parity > 5] <- "6+"
# make Parity column a factor
milkdata$Parity <- as.factor(milkdata$Parity)

# View frequency table for Parity data
kable(t(table(milkdata$Parity)), align = 'c')
```

1	2	3	4	5	6+
145	115	73	40	30	26

The *Date_of_sampling* column contains the date which each observation was recorded. Various different dates are included in the data set, so I have chosen to group dates by month.

The code below identifies the month in the date for each observation, and replaces it with just the number of the month. Again, this has been converted to a factor, and a frequency table of the updated column is shown below.

```r
# Create vector of strings representing month numbers
months <- c("01", "02", "03", "04", "05", "06", "07", "08", "09", "10", "11", "12")

# Loop through each month, replace date with Month
for(MM in months) {
  milkdata$Date_of_sampling[sapply(paste0("/", MM, "/"), grepl, milkdata$Date_of_sampling)] <- MM
}

# make Date_of_sampling column a factor
milkdata$Date_of_sampling <- as.factor(milkdata$Date_of_sampling)

# View frequency table for Date_of_sampling data
kable(t(table(milkdata$Date_of_sampling)), align = 'c')
```

01	02	03	04	06	07	08	09	10
70	26	80	39	47	24	64	56	24

The final categorical variable that will be considered in the data frame is *Milking_Time*. This is a record of whether a milk sample was taken in the morning or in the evening, with values of 1 and 2 corresponding to morning and evening respectively.

For readability, I have replaced the values of 1 and 2 with *Morning* and *Evening*, and I have changed the type of the column to a factor.

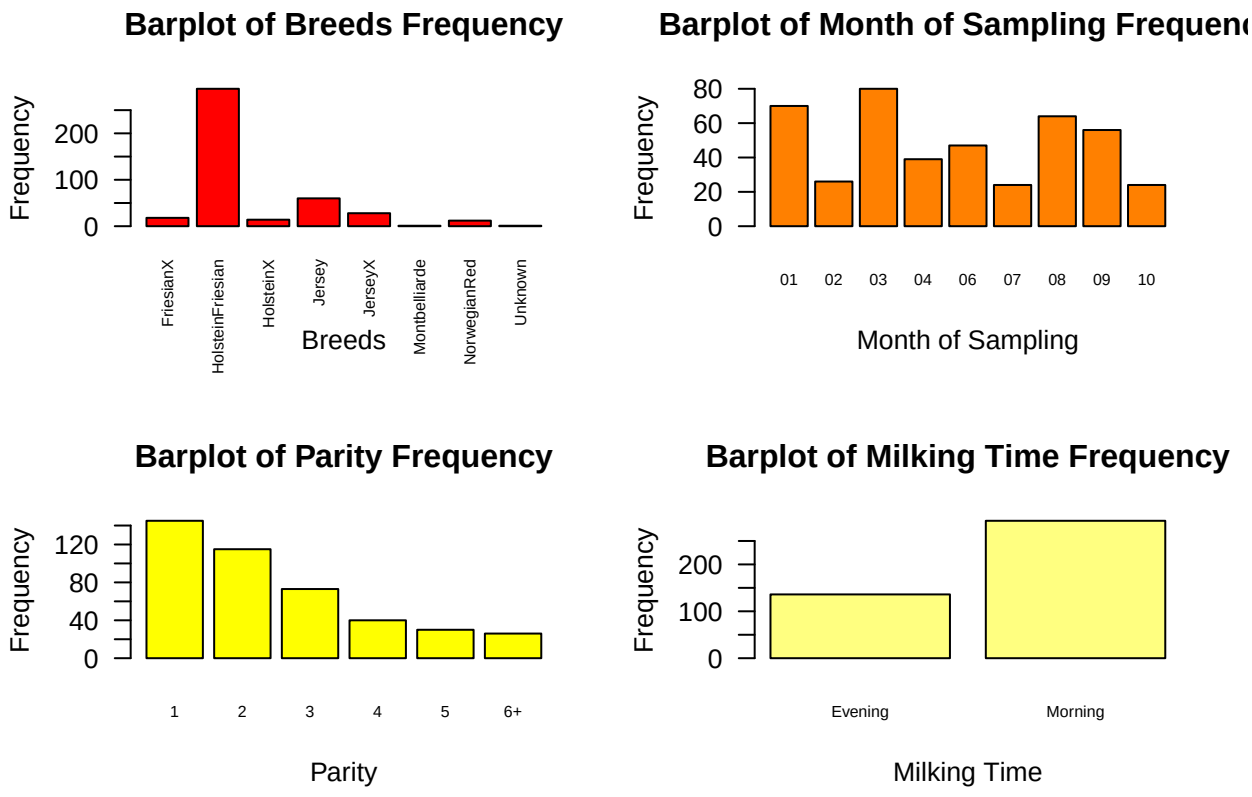
```r
# Set Milking time values of 1 to morning, 2 to evening
milkdata$Milking_Time[milkdata$Milking_Time == 1] <- "Morning"
milkdata$Milking_Time[milkdata$Milking_Time == 2] <- "Evening"

# make Milking_Time column a factor
milkdata$Milking_Time <- as.factor(milkdata$Milking_Time)

# View frequency table for MilkingTime data
```

```
kable(t(table(milkdata$Milking_Time)), align = 'c')
```

Evening	Morning
136	293

Barplots of these 4 categorical traits are plotted below, to show the frequency of each category for each trait.

```
# set plot window to 2 rows x 2 columns
par(mfrow=c(2,2))

# vector of labels for barplots
cats_vec <- c("Breeds", "Month of Sampling", "Parity", "Milking Time")
# vector for label orientation
las_vec <- c(2,1,1,1)

# loop through each categorical trait
for(i in 1:4){
  barplot(table(milkdata[,i]), las=las_vec[i], ylab = "Frequency", xlab = cats_vec[i],
          main = paste("Barplot of", cats_vec[i], "Frequency"), col = heat.colors(4)[i], cex.names = 0.
}
```



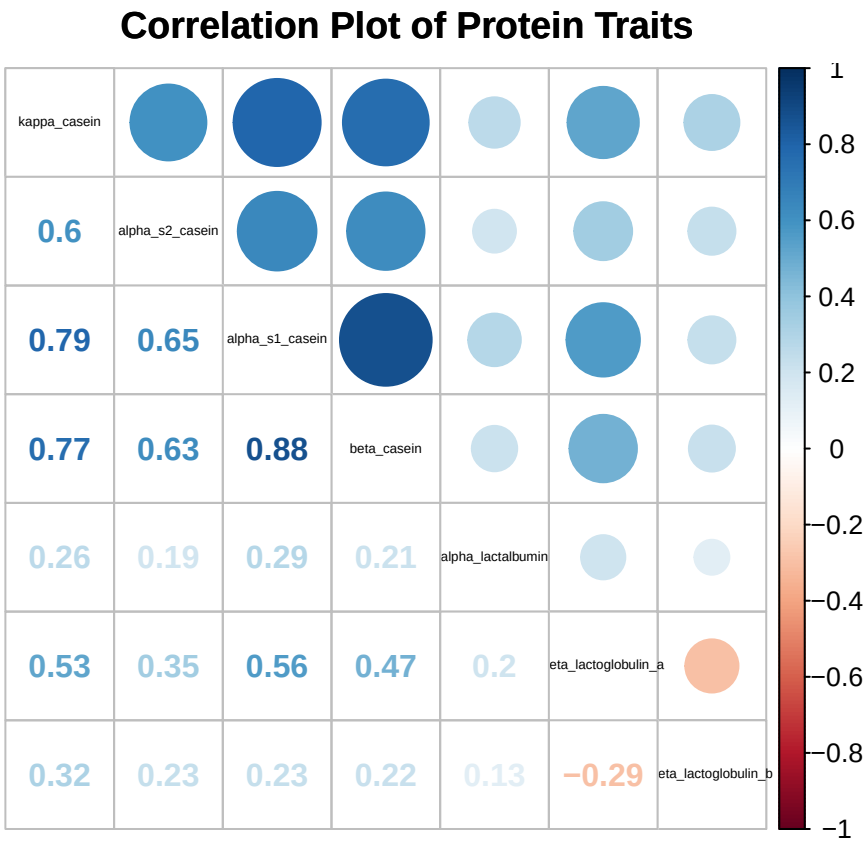
1 (a) Protein Traits Analysis

kappa_casein / alpha_s1_casein / alpha_s2_casein / beta_casein

There are 7 protein traits recorded for each milk sample in the given data. A correlation plot can be used to quickly visualise the strength of the correlation between each variable before looking further into the data.

From the plot, it appears that the first 4 variables (*kappa_casein*, *alpha_s1_casein*, *alpha_s2_casein*, *beta_casein*) are all relatively strongly positively correlated with one another.

```
# Create data frame of protein traits
milk_prot <- milkdata[,5:11]

# Create correlation plot for all protein traits
milk_prot_cor <- cor(na.omit(milk_prot))
corrplot.mixed(milk_prot_cor, tl.cex=0.4, tl.col="black", main = "          Correlation Plot of Protein
```



From inspecting the data, a number of protein values have not been recorded for various observations. I have decided to remove all observations containing *NA* values for the protein traits using the *na.omit* function before proceeding with the analysis of the protein traits. Due to the relatively large number of protein traits to analyse, I will first look at the first 4 protein traits, and will later look at the remaining 3 protein traits separately.

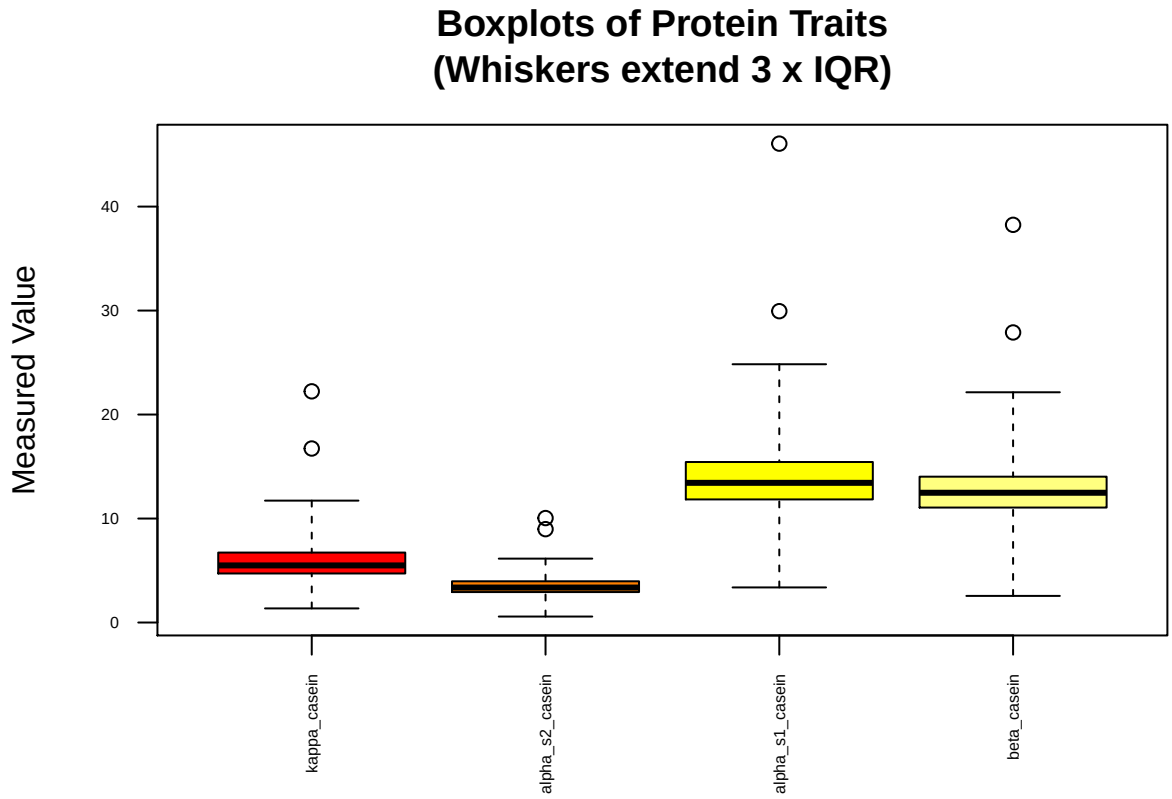
Boxplots are a useful tool for visualising the distribution of numerical data, displaying the median, first and third quartiles for the data, while also highlighting any data points that could be considered as outliers.

There are numerous ways that we can classify data points as outliers, such as points with values outside 3 standard deviations from the mean of the data, or points that are a specified distance above the third quartile or below the first quartile of the data.

For this project, I have chosen to look at data points that are greater than 3 times the interquartile range above the third quartile, and 3 times the interquartile range below the first quartile, when considering values that may be genuine outliers in the data that should be considered for removal. In the boxplots below, I have defined the whiskers to extend this length, so that any data points that could be considered as extreme outliers can easily be identified on the plots.

The first boxplot below shows the distribution of the first 4 protein traits in their original scale. For each trait there are 2 points shown outside the range of the whiskers, which could be considered extreme outliers.

```
# Drop rows that include NAs
milk_prot <- na.omit(milk_prot)
rownames(milk_prot) <- 1:nrow(milk_prot)

# boxplot
boxplot(milk_prot[,1:4], range=3, las=2, cex.axis=0.5, col=heat.colors(4), main="Boxplots of Protein Tra
```



To determine exactly which observations these outliers are from, a different boxplot function from the *car* library can be used to identify the index of these observations.

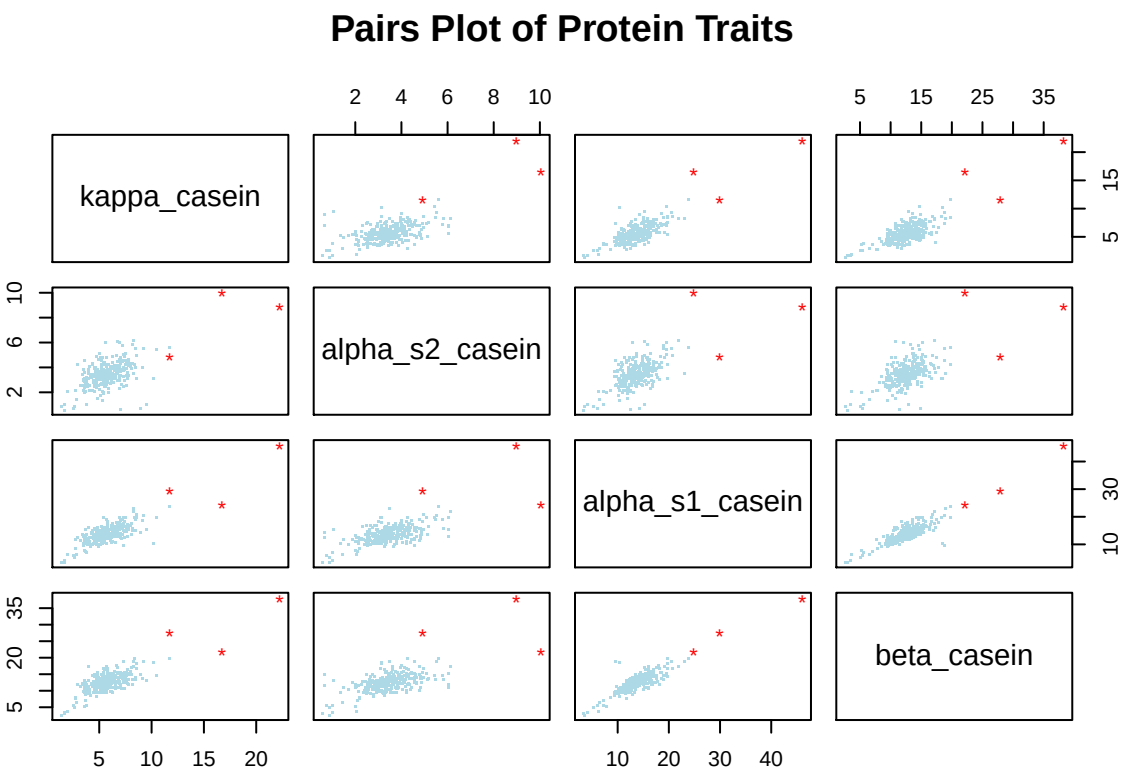
Also, it can be difficult at times to compare different distributions on the same boxplot if their ranges and distributions differ significantly, so it can be useful to plot the standardised data distributions on the boxplot. As a result, each trait in the boxplot below has a mean value of 0 and standard deviation value of 1, and the distributions all have a similar scale.

```
# Boxplot of first 4 protein traits
bp1 <- Boxplot(scale(milk_prot[,1:4]), pch="*", range=3,
        id=list(cex=0.5, location="avoid", col="red"),
        las=2, cex.axis=0.5, col=heat.colors(4),
        main="Boxplots of Standardised Protein Traits\n(Whiskers extend 3 x IQR)", ylab="Standardised Va
```

From the boxplot above, observations 69, 71 and 79 all appear to be outliers considering the values recorded for some of the protein traits.

To further visualise these outliers, we can look at pairs plots of the first 4 protein traits, plotting the relationship between each pair of traits. In the pairs plots below, the 3 outliers identified on the boxplot are highlighted in red. Again, these observations appear to be legitimate outliers, clearly deviating significantly from the bulk of the data in most of the plots below.

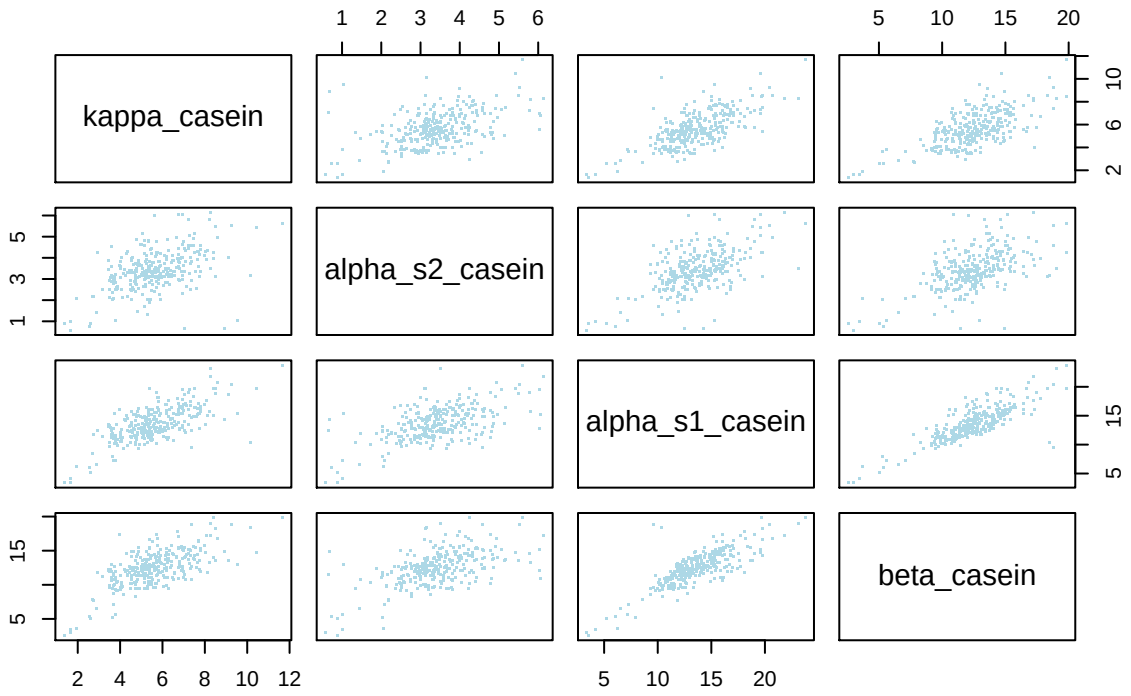
```
# get indices of outliers
inds <- unique(as.numeric(bp1))

# create vector of 1s and 2s corresponding to outliers
colvec <- rep(1, nrow(milk_prot))
colvec[inds] <- 2

# Pairs plot of the first 4 protein traits
pairs(milk_prot[,1:4], pch=c(".", "*")[colvec], col=c("lightblue", "red")[colvec], main="Pairs Plot of
```



Without any further knowledge available regarding why these observations have such extreme values for some of the recorded protein traits, I have decided to omit them from the data when calculating the summary statistics for these traits.

Below, an updated pairs plot is shown with outlier observations removed to get a better visualisation of relationships between the traits, and a table of summary statistics for these traits with the identified outliers removed. This includes a Shapiro Wilk W test statistic value for each variable, which is a method of measuring how close a distribution is to normality.

```
# Updated Pairs plot of the first 4 protein traits
pairs(milk_prot[-inds,1:4], pch=".", col="lightblue", main="Pairs Plot of Protein Traits")
```



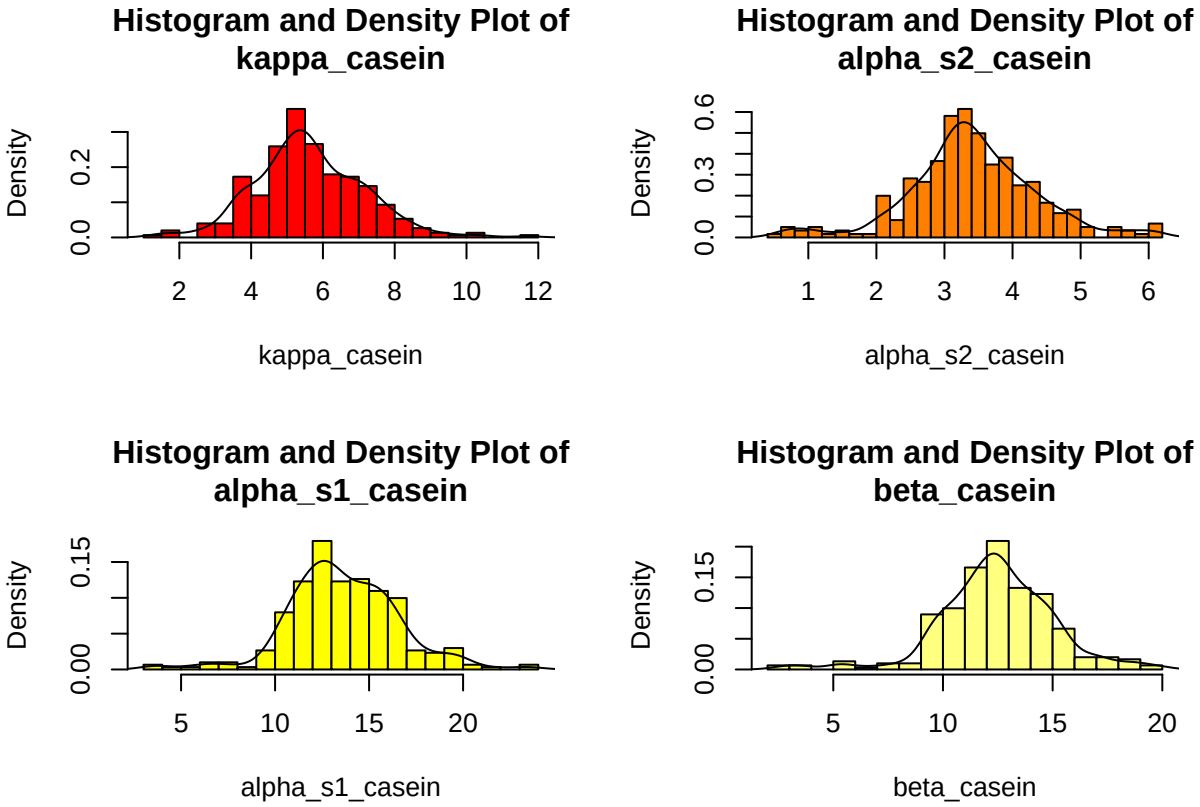

```
# get summary stats of reduced dataset
sumtab1 <- as.data.frame(apply(milk_prot[-inds, 1:4], 2, summary))

# get stand.dev. values for reduced dataset
sds1 <-apply(milk_prot[-inds, 1:4], 2, sd)

# get shapiro wilkes test statistic values for reduced dataset (normality test)
sw1 <- NULL
# loop through indices of relevant columns
for(i in 1:4) {
  # calculate W value for the current column
  sw1 <- c(shapiro.test(milk_prot[-inds,i])$statistic, sw1)
}

# bind stand. deviations and shapiro wilk W values to summary data frame
sumtab1 <- round(rbind(sumtab1, sds1, sw1), 3)
rownames(sumtab1)[7:8] <- c("St.Dev", "W")

# print table of summary statistics
kable(sumtab1, align = 'c')
```

	kappa_casein	alpha_s2_casein	alpha_s1_casein	beta_casein
Min.	1.357	0.576	3.377	2.559
1st Qu.	4.716	2.928	11.825	11.057
Median	5.472	3.357	13.418	12.457
Mean	5.626	3.406	13.626	12.443
3rd Qu.	6.641	3.938	15.373	14.019
Max.	11.671	6.146	23.759	19.833

	kappa_casein	alpha_s2_casein	alpha_s1_casein	beta_casein
St.Dev	1.513	0.946	2.942	2.593
W	0.964	0.976	0.976	0.987

We can also visualise the distributions of each of the traits using histograms as shown below. The outliers identified above have been omitted when creating these plots.

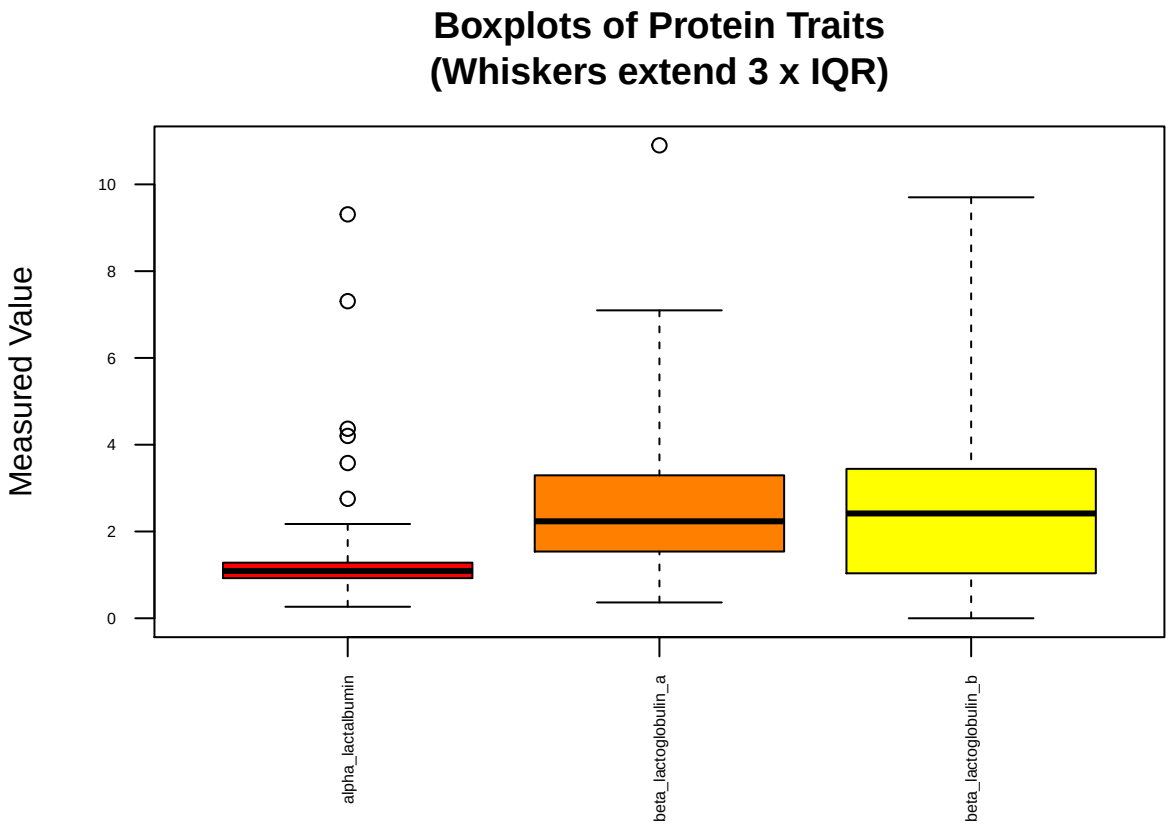
```
# set plot window to 2 rows x 2 columns
par(mfrow=c(2,2))

# loop through each trait
for(i in 1:4){
  # Plot probability density histogram of selected trait
  hist(milk_prot[-inds,i], freq = FALSE, breaks = 20,
       main = paste0("Histogram and Density Plot of\n", names(milk_prot)[i]),
       xlab = names(milk_prot)[i], col = heat.colors(4)[i])

  # plot density plot of selected trait
  lines(density(milk_prot[-inds,i]))
}
```



alpha_lactalbumin / beta_lactoglobulin_a / beta_lactoglobulin_b

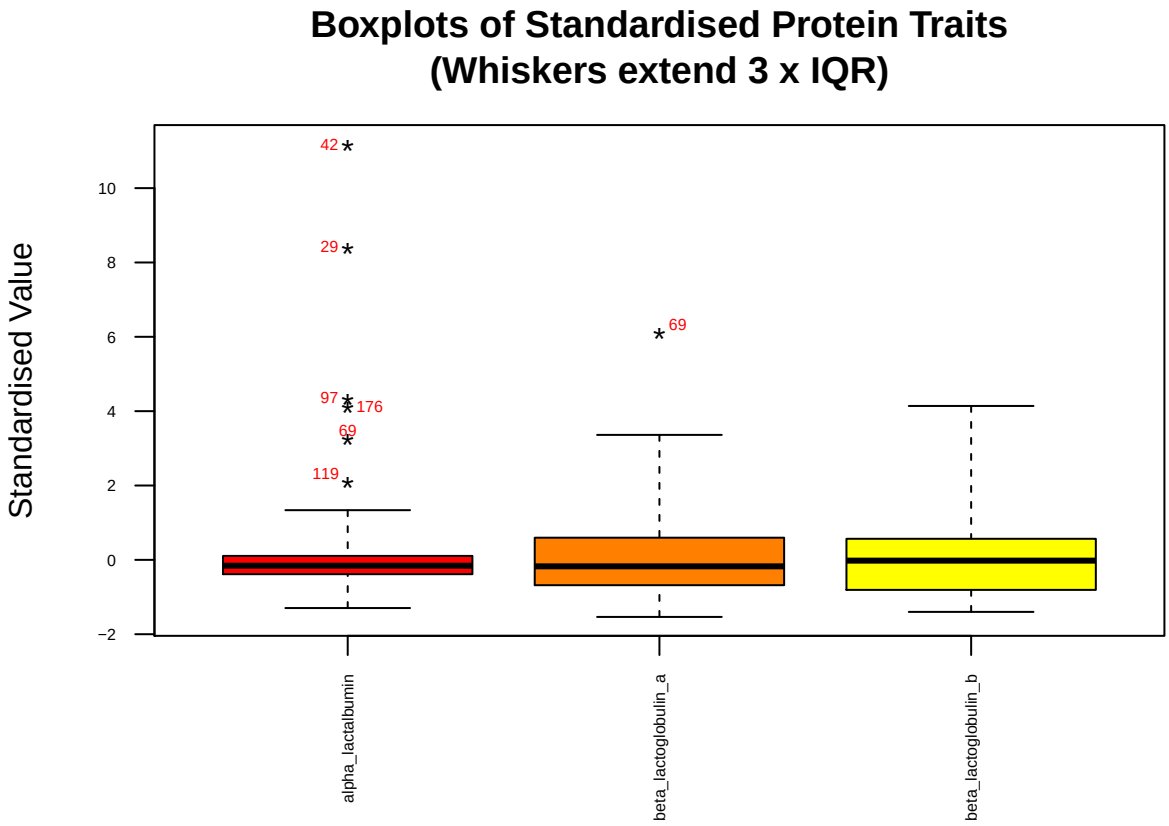
The boxplot below shows the distribution of the final 3 protein traits in their original scale. For the *alpha_lactalbumin* trait in particular, there are a number of points shown outside the range of the whiskers, highlighting them as potential extreme outliers in the data.

```
# boxplot
boxplot(milk_prot[,5:7], range=3, las=2, cex.axis=0.5, col=heat.colors(4), main="Boxplots of Protein Tra
```



Again, we can plot the observation index of the identified outliers on the boxplot, and we can also standardise each of the traits to make the distributions of each trait easier to visualise when shown side by side on the same plot.

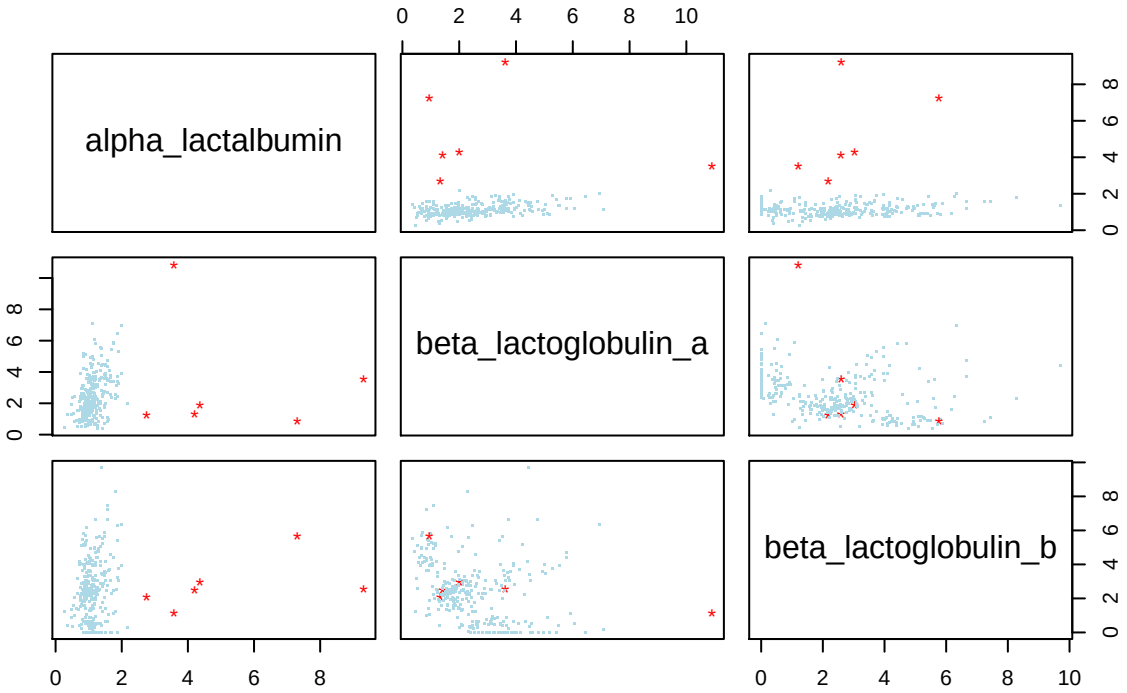
```
# Boxplot of first 4 protein traits
bp2 <- Boxplot(scale(milk_prot[,5:7]), pch="*", range=3,
        id=list(cex=0.5, location="avoid", col="red"),
        las=2, cex.axis=0.5, col=heat.colors(4),
        main="Boxplots of Standardised Protein Traits\n(Whiskers extend 3 x IQR)", ylab="Standardised Va
```



From the boxplot above, 6 different observations were identified as potential outliers considering the values recorded for some of the protein traits.

To further visualise these outliers, we can look at pairs plots of the final 3 protein traits, plotting the relationship between each pair of traits. In the pairs plots below, the outliers identified on the boxplot are highlighted in red. Again, these appear to be legitimate outliers, clearly deviating significantly from the bulk of the data in some of the plots below.

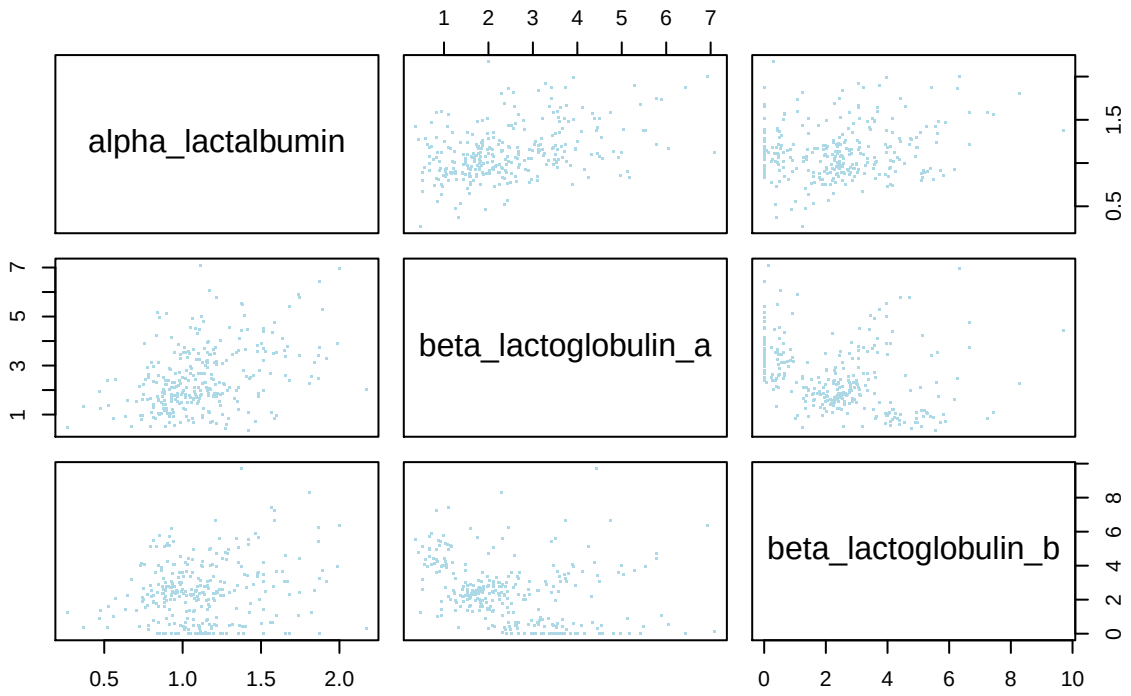
```
# get indices of outliers
inds2 <- unique(as.numeric(bp2))

# create vector of 1s and 2s corresponding to outliers
colvec <- rep(1, nrow(milk_prot))
colvec[inds2] <- 2

# Pairs plot of the first 4 protein traits
pairs(milk_prot[,5:7], pch=c(".", "*")[colvec], col=c("lightblue", "red")[colvec], main="Pairs Plot of
```



Aside from the outliers, another thing that can be observed from the pairs plots is that several observations in the data have a *beta_lactoglobulin_b* value of exactly zero. While this seems unlikely, there are a significant number of observations with *beta_lactoglobulin_b* values that are very close to zero, so there is a possibility that the values measured were very close to zero and were recorded as such. Without further knowledge available, I have chosen not to exclude these observations from the analysis.

Conversely, without any further knowledge available on the identified outliers for these protein traits, I have decided to omit them from the data when calculating the summary statistics for these traits.

```
# Updated Pairs plot of the first 4 protein traits
pairs(milk_prot[-inds2,5:7], pch=".", col="lightblue", main="Pairs Plot of Protein Traits")
```



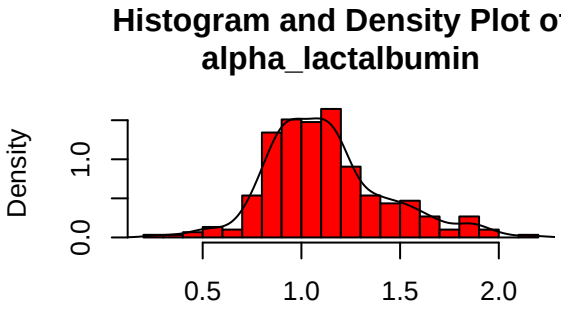

```
# get summary stats of reduced dataset
sumtab2 <- as.data.frame(apply(milk_prot[-inds2, 5:7], 2, summary))

# get stand.dev. values for reduced dataset
sds2 <-apply(milk_prot[-inds2, 5:7], 2, sd)

# get shapiro wilkes test statistic values for reduced dataset (normality test)
sw2 <- NULL
# loop through indices of relevant columns
for(i in 5:7) {
  # calculate W value for the current column
  sw2 <- c(shapiro.test(milk_prot[-inds2,i])$statistic, sw2)
}

# bind stand. deviations and shapiro wilk W values to summary data frame
sumtab2 <- rbind(sumtab2, sds2, sw2)
rownames(sumtab2)[7:8] <- c("St.Dev", "W")

# print table of summary statistics
kable(sumtab2, align = 'c')
```

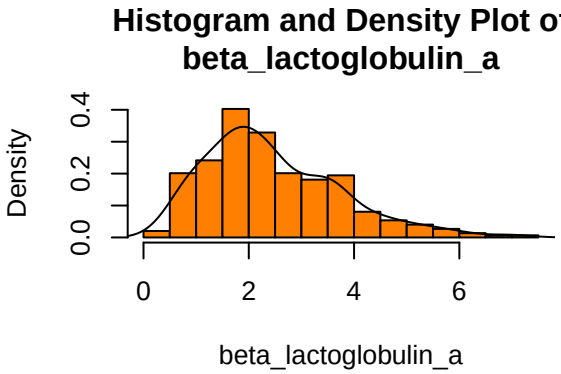
	alpha_lactalbumin	beta_lactoglobulin_a	beta_lactoglobulin_b
Min.	0.2647158	0.3640380	0.0000000
1st Qu.	0.9223530	1.5513175	0.9641435
Median	1.0799210	2.2379770	2.4078680
Mean	1.1225908	2.4564317	2.4421538
3rd Qu.	1.2561535	3.2785950	3.4535215
Max.	2.1724260	7.0960000	9.7015400

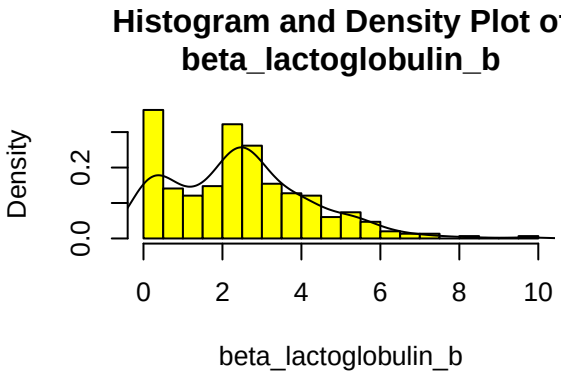
	alpha_lactalbumin	beta_lactoglobulin_a	beta_lactoglobulin_b
St.Dev	0.3020304	1.2913684	1.7563867
W	0.9518432	0.9473425	0.9638243

We can also visualise the distributions of each of the traits using histograms as shown below. The outliers identified above have been omitted when creating these plots.

```
# set plot window to 2 rows x 2 columns
par(mfrow=c(2,2))

# loop through each trait
for(i in 5:7){
  # Plot probability density histogram of selected trait
  hist(milk_prot[-inds2, i], freq = FALSE, breaks = 20,
       main = paste0("Histogram and Density Plot of\n", names(milk_prot)[i]),
       xlab = names(milk_prot)[i], col = heat.colors(3)[i-4])

  # plot density plot of selected trait
  lines(density(milk_prot[-inds2, i]))
}
```





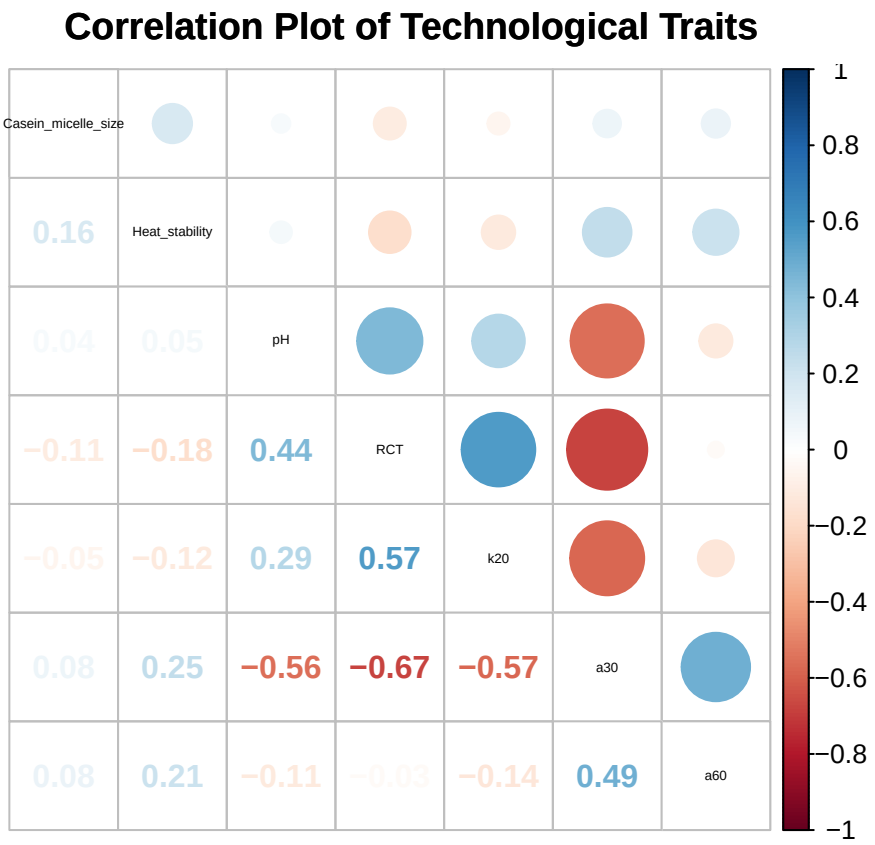


1 (b) Technological Traits Analysis

Casein_micelle_size / Heat_stability / pH

There are 7 technological traits recorded for each milk sample in the given data. A correlation plot can be used to quickly visualise the strength of the correlation between each variable before looking further into the data.

```
# Create data frame of technological traits
milk_tech <- milkdata[,12:18]

# Create correlation plot for all protein traits
milk_tech_cor <- cor(na.omit(milk_tech))
corrplot.mixed(milk_tech_cor, tl.cex=0.4, tl.col="black",
               main = "         Correlation Plot of Technological Traits", mar = c(0,0,2,0))
```



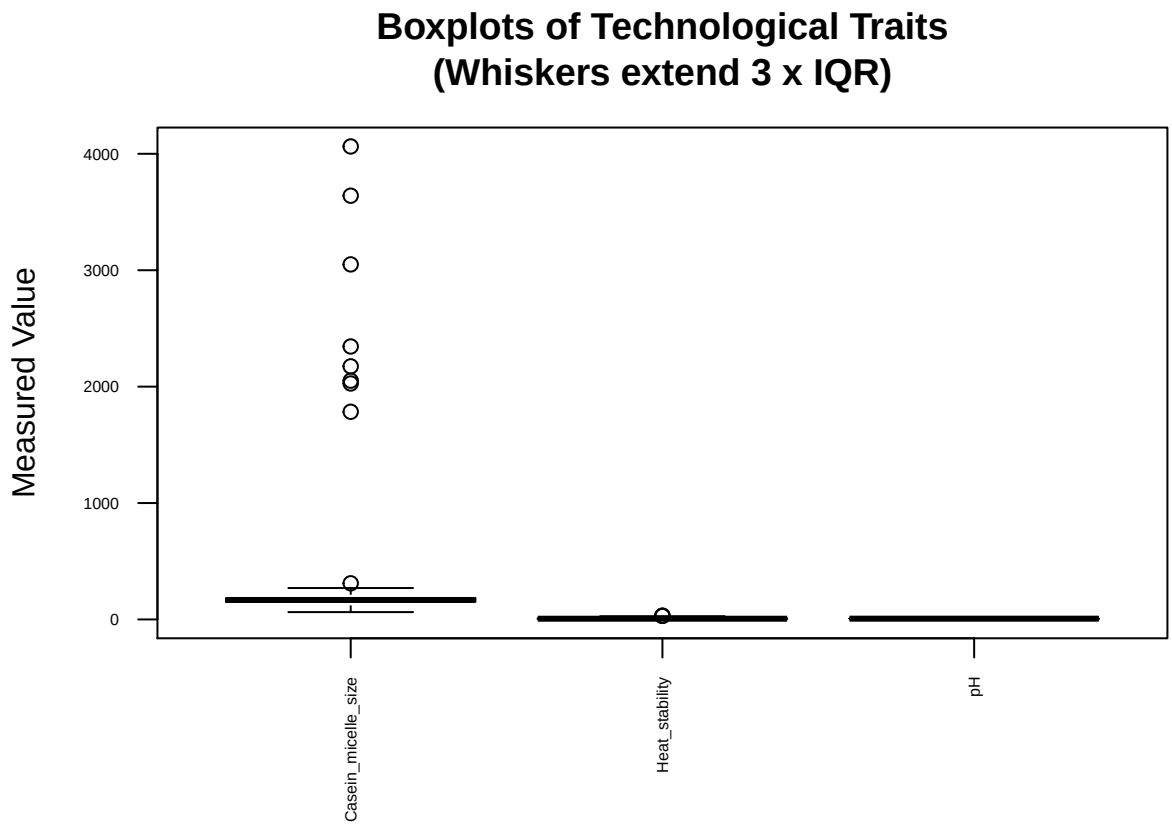
Like what we saw for the protein traits, there are several cases where values have not been recorded for some technological traits various observations. I have decided to remove all observations containing *NA* values for the technological traits using the *na.omit* function before proceeding with the analysis of the technological traits.

Due to the relatively large number of technological traits to analyse, I will first look at the first 3 traits, and will later look at the remaining 4 traits traits separately.

As stated earlier, I have chosen to look at data points that are greater than 3 times the interquartile range above the third quartile, and 3 times the interquartile range below the first quartile, when considering values that may be genuine outliers in the data that should be considered for removal. In the boxplots below, I have defined the whiskers to extend this length, so that any data points that could be considered as extreme outliers can easily be identified on the plots.

The first boxplot below shows the distribution of the first 3 technological traits in their original scale.

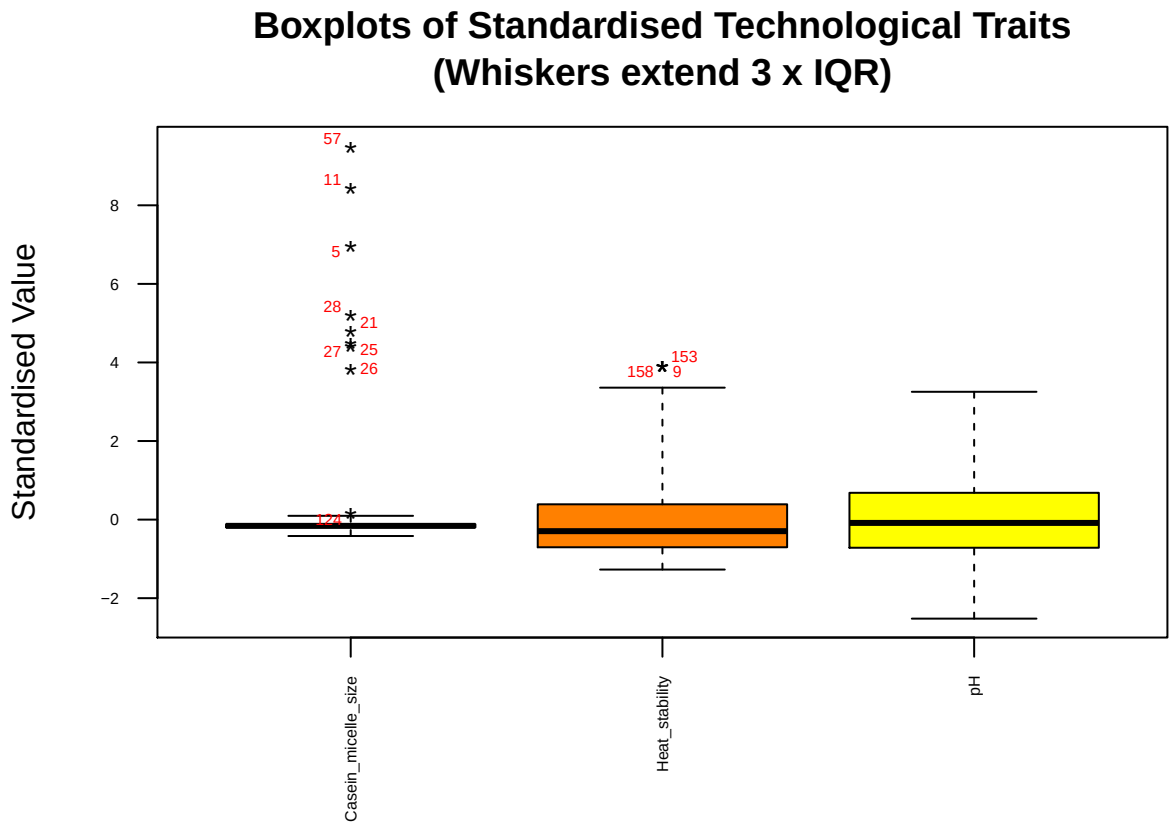
```
# Drop rows that include NAs
milk_tech <- na.omit(milk_tech)
rownames(milk_tech) <- 1:nrow(milk_tech)

# boxplot of first 3 tech traits
boxplot(milk_tech[,1:3], range=3, las=2, cex.axis=0.5, col=heat.colors(3), main="Boxplots of Technologi
```



To determine exactly which observations these outliers are from, a different boxplot function from the *car* library can be used to identify the index of these observations.

Due to the large difference in the magnitude of the ranges in each of these distributions, it is difficult to visualise each distribution on the same boxplot in their original scale. Therefore, I have also decided to show the standardised data distributions on the boxplot.

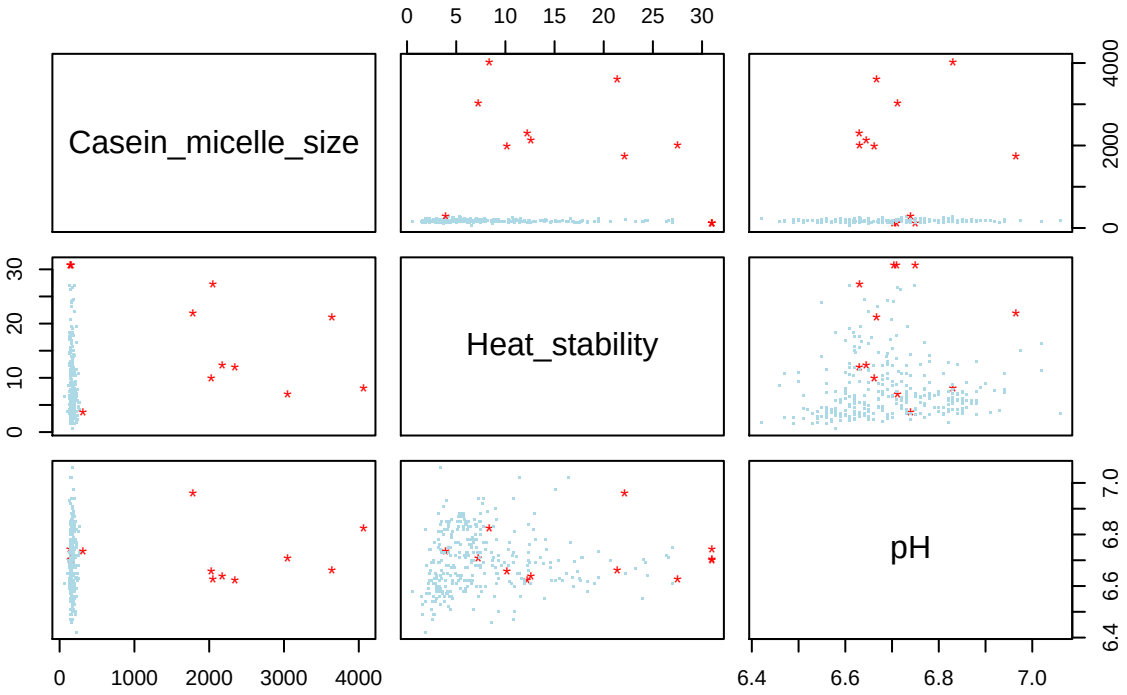
```
# Boxplot of first 3 tech traits
bp3 <- Boxplot(scale(milk_tech[,1:3]), pch="*", range=3,
        id=list(cex=0.5, location="avoid", col="red"),
        las=2, cex.axis=0.5, col=heat.colors(3),
        main="Boxplots of Standardised Technological Traits\n(Whiskers extend 3 x IQR)", ylab="Standardi
```



From the boxplot above, several observations were identified as potential outliers considering the values recorded for some of the technological traits, in particular for *Casein_micelle_size*.

To further visualise these outliers, we can look at pairs plots of the first 3 technological traits, plotting the relationship between each pair of traits. In the pairs plots below, the outliers identified on the boxplot are highlighted in red. Again, the majority of these observations appear to be legitimate outliers, clearly deviating significantly from the bulk of the data in most of the plots below.

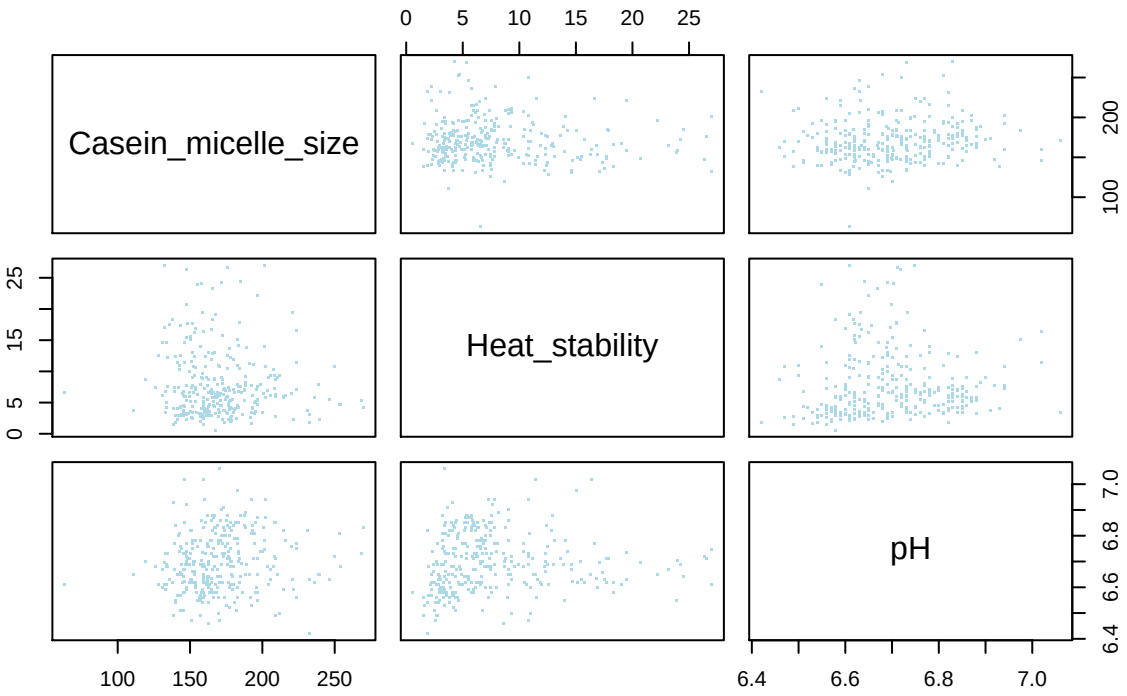
```
# get indices of outliers
inds3 <- unique(as.numeric(bp3))

# create vector of 1s and 2s corresponding to outliers
colvec <- rep(1, nrow(milk_tech))
colvec[inds3] <- 2

# Pairs plot of the first 4 protein traits
pairs(milk_tech[,1:3], pch=c(".", "*")[colvec], col=c("lightblue", "red")[colvec], main="Pairs Plot of
```



Several of the outliers relating to the *Casein_micelle_size* trait appear to be an order of magnitude away from the bulk of the data. It is possible that these values may have been entered correctly (i.e. decimal point omitted), but without being able to confirm this, I have decided that these observations should be omitted.

Without any further knowledge available regarding why these observations have such extreme values for some of the recorded protein traits, I have decided to omit them from the data when calculating the summary statistics for these traits.

```
# Updated Pairs plot of the first 4 protein traits
pairs(milk_tech[-inds3, 1:3], pch=".", col="lightblue", main="Pairs Plot of Technological Traits")
```



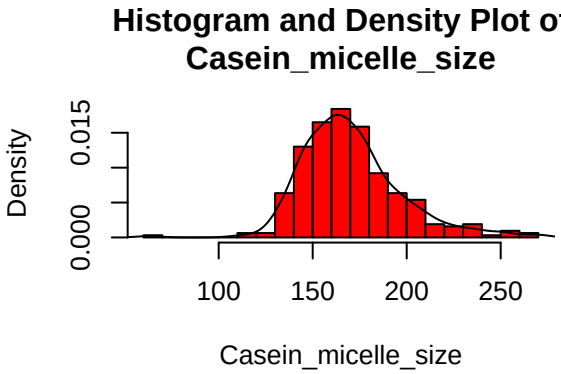

```
# get summary stats of reduced dataset
sumtab3 <- as.data.frame(apply(milk_tech[-inds3, 1:3], 2, summary))

# get stand.dev. values for reduced dataset
sds3 <- apply(milk_tech[-inds3, 1:3], 2, sd)

# get shapiro wilkes test statistic values for reduced dataset (normality test)
sw3 <- NULL
# loop through indices of relevant columns
for(i in 1:3) {
  # calculate W value for the current column
  sw3 <- c(shapiro.test(milk_tech[-inds3, i])$statistic, sw3)
}

# bind stand. deviations and shapiro wilk W values to summary data frame
sumtab3 <- rbind(sumtab3, sds3, sw3)
rownames(sumtab3)[7:8] <- c("St.Dev", "W")

# print table of summary statistics
kable(sumtab3, align = 'c')
```

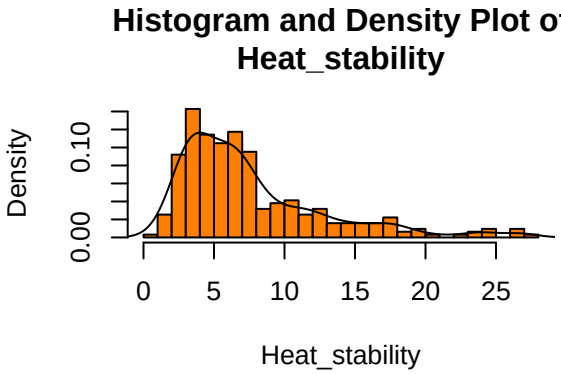
	Casein_micelle_size	Heat_stability	pH
Min.	63.1200000	0.5800000	6.4200000
1st Qu.	151.9500000	3.8150000	6.6170000
Median	166.8000000	6.1700000	6.6900000
Mean	170.4692698	7.5917778	6.6985714
3rd Qu.	183.1000000	9.3100000	6.7800000
Max.	269.9000000	27.0200000	7.0600000

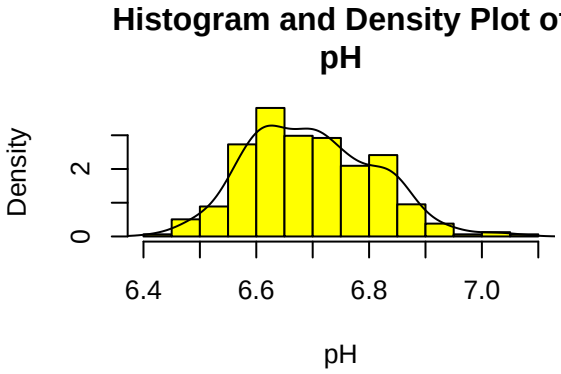
	Casein_micelle_size	Heat_stability	pH
St.Dev	26.5900643	5.2281226	0.1114271
W	0.9876646	0.8437934	0.9481062

We can also visualise the distributions of each of the traits using histograms as shown below. The outliers identified above have been omitted when creating these plots.

```
# set plot window to 2 rows x 2 columns
par(mfrow=c(2,2))

# loop through each trait
for(i in 1:3){
  # Plot probability density histogram of selected trait
  hist(milk_tech[-inds3,i], freq = FALSE, breaks = 20,
       main = paste0("Histogram and Density Plot of\n", names(milk_tech)[i]),
       xlab = names(milk_tech)[i], col = heat.colors(3)[i])

  # plot density plot of selected trait
  lines(density(milk_tech[-inds3,i]))
}
```



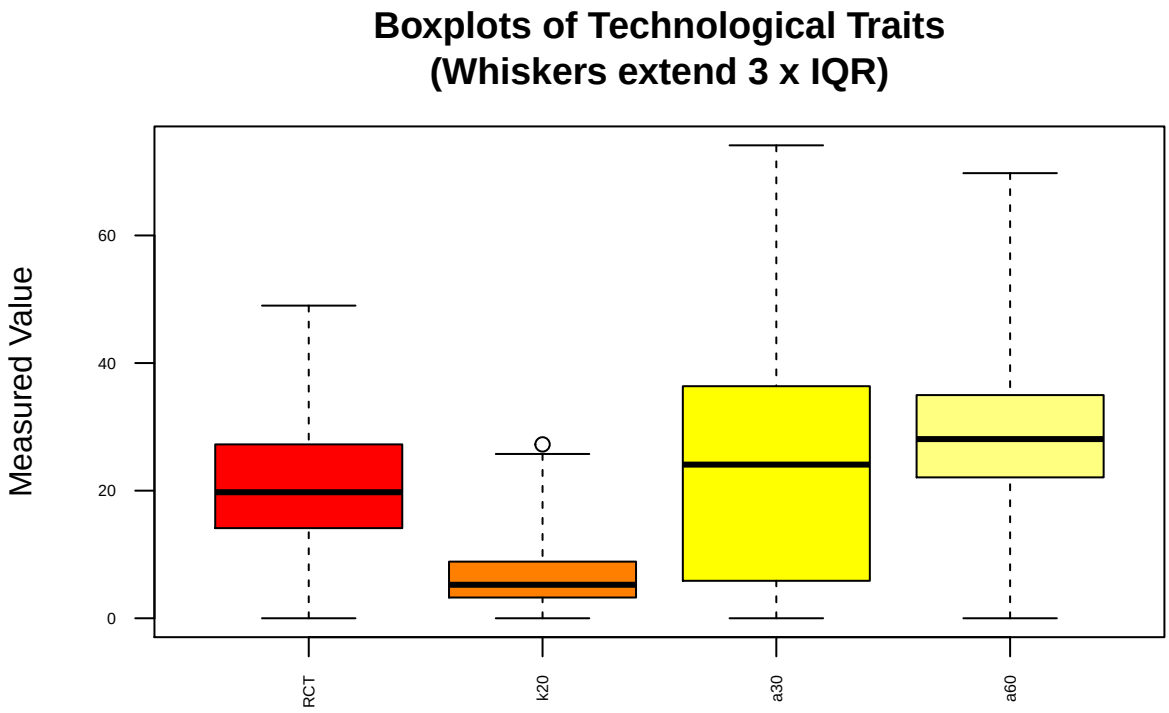




RCT / k20 / a30 / a60

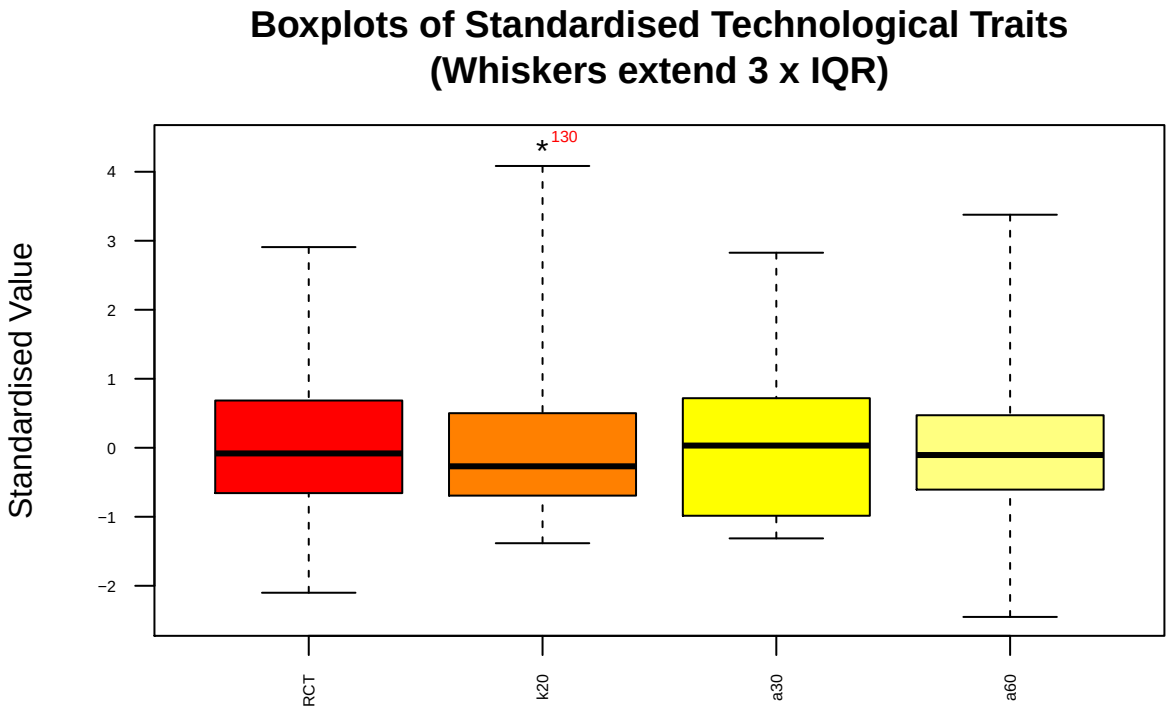
The boxplot below shows the distribution of the final 4 technological traits in their original scale. Only one data point is considered an outlier on the boxplot below (for the *k20* trait) based on my earlier defined criteria.

```
# boxplot of last 4 technological traits
boxplot(milk_tech[,4:7], range=3, las=2, cex.axis=0.5, col=heat.colors(4), main="Boxplots of Technologi
```



Again, we can plot the observation index of the identified outlier on the boxplot, and we can also standardise each of the traits to make the distributions of each trait easier to visualise when shown side by side on the same plot.

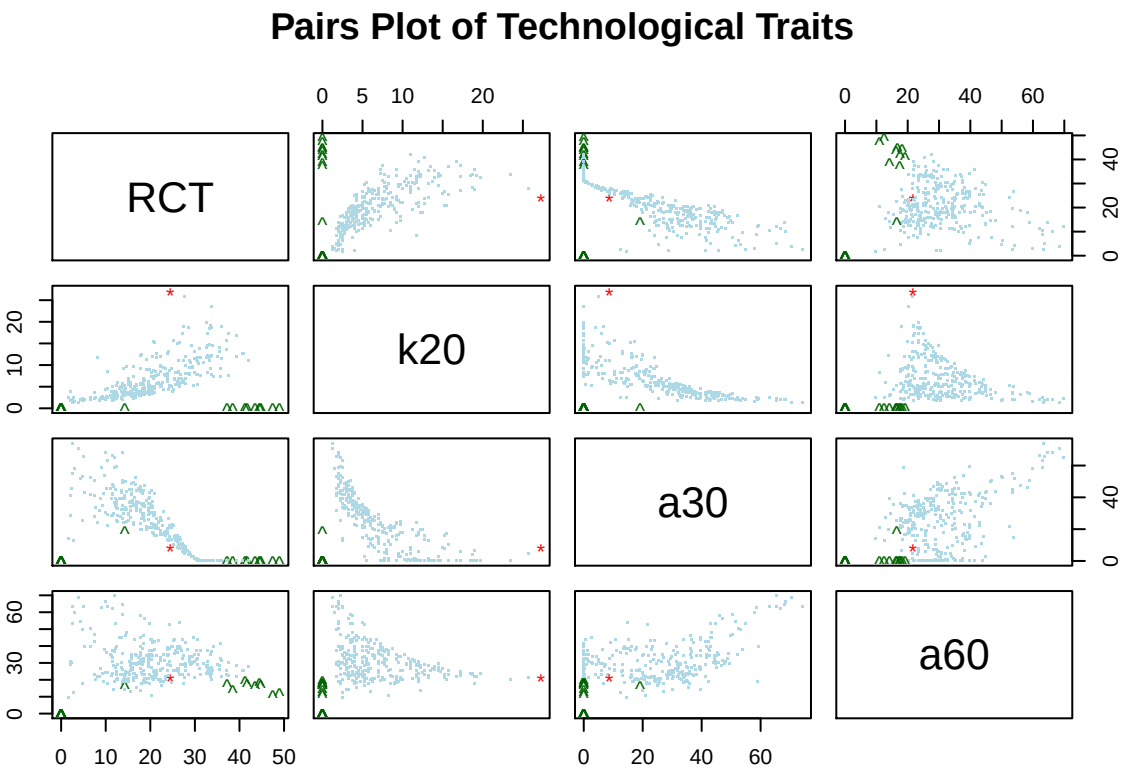
```
# Boxplot of last 4 tech traits
bp4 <- Boxplot(scale(milk_tech[,4:7]), pch="*", range=3,
        id=list(cex=0.5, location="avoid", col="red"),
        las=2, cex.axis=0.5, col=heat.colors(4),
        main="Boxplots of Standardised Technological Traits\n(Whiskers extend 3 x IQR)", ylab="Standard
```



To further visualise the data and the outlier identified above, we can look at pairs plots of the final 4 technological traits, plotting the relationship between each pair of traits. In the pairs plots below, the outlier identified on the boxplot is highlighted in red. Again, it appears to be a legitimate outlier, clearly deviating significantly from the bulk of the data in some of the plots below.

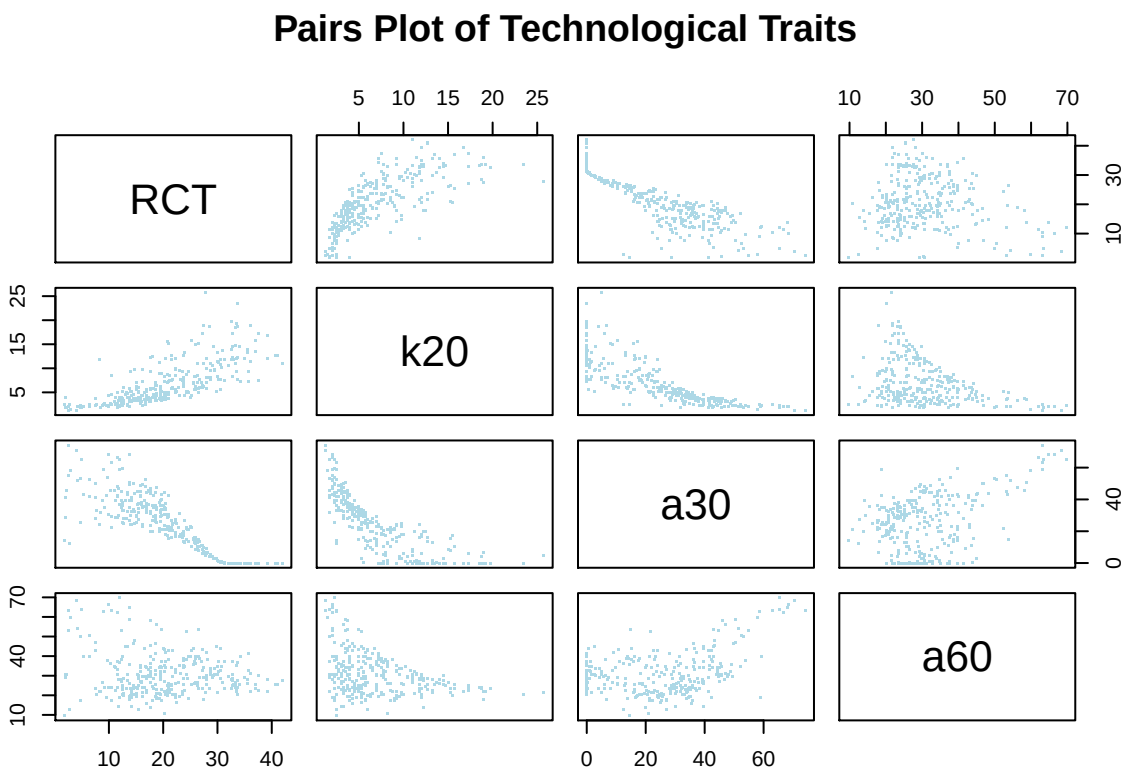
It is also apparent from looking at the pairs plots that a number of observations have zero values recorded for some of the technological traits, most noticeably in the *k20* (curd firming time) and *a30* (curd firmness after 30 minutes) traits. While *a30* values of zero appear to be legitimate recordings and follow the trend of the data (e.g. *a30* values appear to decrease linearly with increase in *RCT* until a certain *threshold*RCT* value is reached, at which point only values of zero are measured for *a30*), it doesn't appear that the *k20* values recorded as zero follow the trend of the remaining data. I have plotted observations with zero values for *k20* in green on the pairs plots below for clarity.

```
# get indices of outliers
inds4 <- unique(as.numeric(bp4))

# create vector of 1s and 2s corresponding to outliers
colvec <- rep(1, nrow(milk_tech))
colvec[inds4] <- 2
colvec[which(milk_tech[,5]==0)] <- 3

# Pairs plot of the last 4 tech traits
pairs(milk_tech[,4:7], pch=c(".", "*", "^")[colvec], col=c("lightblue", "red", "darkgreen")[colvec], ma
```



Without any further knowledge available on the identified outliers or the zero values recorded for *k20* for some observations for these technological traits, I have decided to omit them from the data when calculating the summary statistics for these traits.

```r
# Updated Pairs plot of the last 4 protein traits
pairs(milk_tech[-c(inds4, which(milk_tech[,5]==0)), 4:7], pch=".", col="lightblue", main="Pairs Plot of
```



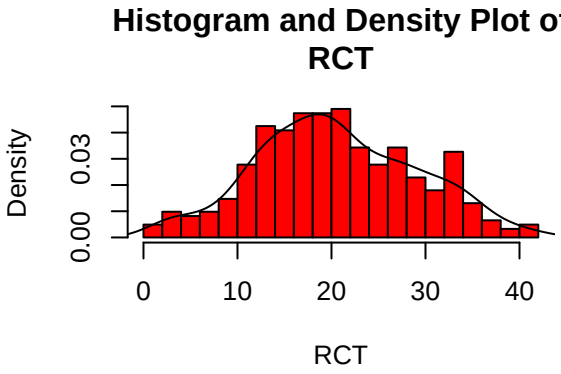

```r
# get summary stats of reduced dataset
sumtab4 <- as.data.frame(apply(milk_tech[-c(inds4, which(milk_tech[,5]==0)), 4:7], 2, summary))

# get stand.dev. values for reduced dataset
sds4 <- apply(milk_tech[-c(inds4, which(milk_tech[,5]==0)), 4:7], 2, sd)

# get shapiro wilkes test statistic values for reduced dataset (normality test)
sw4 <- NULL
# loop through indices of relevant columns
for(i in 4:7) {
  # calculate W value for the current column
  sw4 <- c(shapiro.test(milk_tech[-c(inds4, which(milk_tech[,5]==0)), i])$statistic, sw4)
}

# bind stand. deviations and shapiro wilk W values to summary data frame
sumtab4 <- rbind(sumtab4, sds4, sw4)
rownames(sumtab4)[7:8] <- c("St.Dev", "W")

# print table of summary statistics
kable(sumtab4, align = 'c')
```

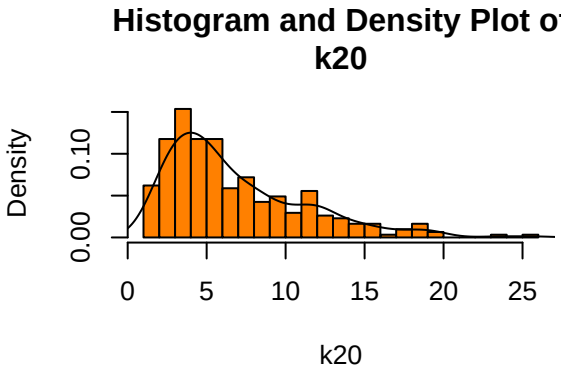
	RCT	k20	a30	a60
Min.	1.7500000	1.2500000	0.0000000	9.6000000
1st Qu.	14.5000000	3.7500000	10.3250000	23.3000000

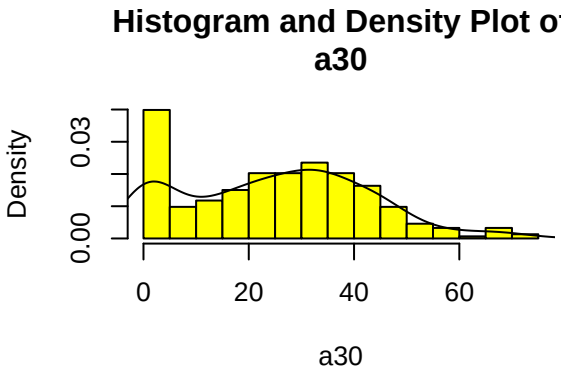
	RCT	k20	a30	a60
Median	19.7500000	5.5000000	25.5500000	28.8000000
Mean	20.5678105	6.8733660	25.0330719	30.7601961
3rd Qu.	26.6875000	9.2500000	36.7200000	35.4100000
Max.	42.0000000	25.7500000	74.1200000	69.7600000
St.Dev	8.5342186	4.4034843	17.4659741	10.7989965
W	0.9152271	0.9572039	0.8875998	0.9898723

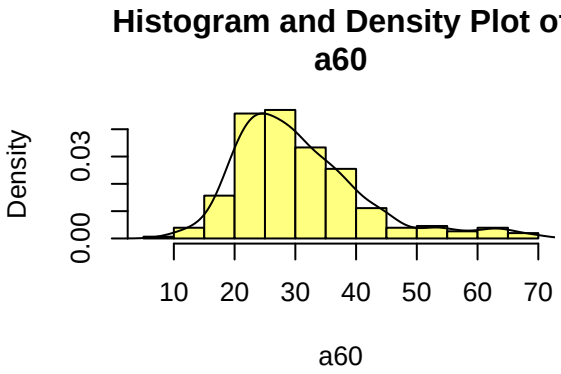
We can also visualise the distributions of each of the traits using histograms as shown below. The outliers identified above have been omitted when creating these plots.

```
# set plot window to 2 rows x 2 columns
par(mfrow=c(2,2))

# loop through each trait
for(i in 4:7){
  # Plot probability density histogram of selected trait
  hist(milk_tech[-c(which(milk_tech[,5]==0), inds4), i], freq = FALSE, breaks = 20,
       main = paste0("Histogram and Density Plot of\n", names(milk_tech)[i]),
       xlab = names(milk_tech)[i], col = heat.colors(4)[i-3])

  # plot density plot of selected trait
  lines(density(milk_tech[-c(which(milk_tech[,5]==0), inds4), i]))
}
```









2. Clusters

We will now look at the MIR spectra data in the given data set, and we will look for clusters of similar spectra. Firstly, we can create a new data frame with all columns removed except for the spectra data.

Three separate ranges of wavenumbers are provided in the data set: 941cm^{-1} - 1601cm^{-1}, 1736cm^{-1} - 3005cm^{-1} and 3714cm^{-1} - 3815cm^{-1}. For the purposes of clear and accurate plotting of the spectra data, I will create vector of the given wavenumbers (by dropping the "X" from the column names) and I will create 3 separate vectors of column index values corresponding to each of the ranges defined above.

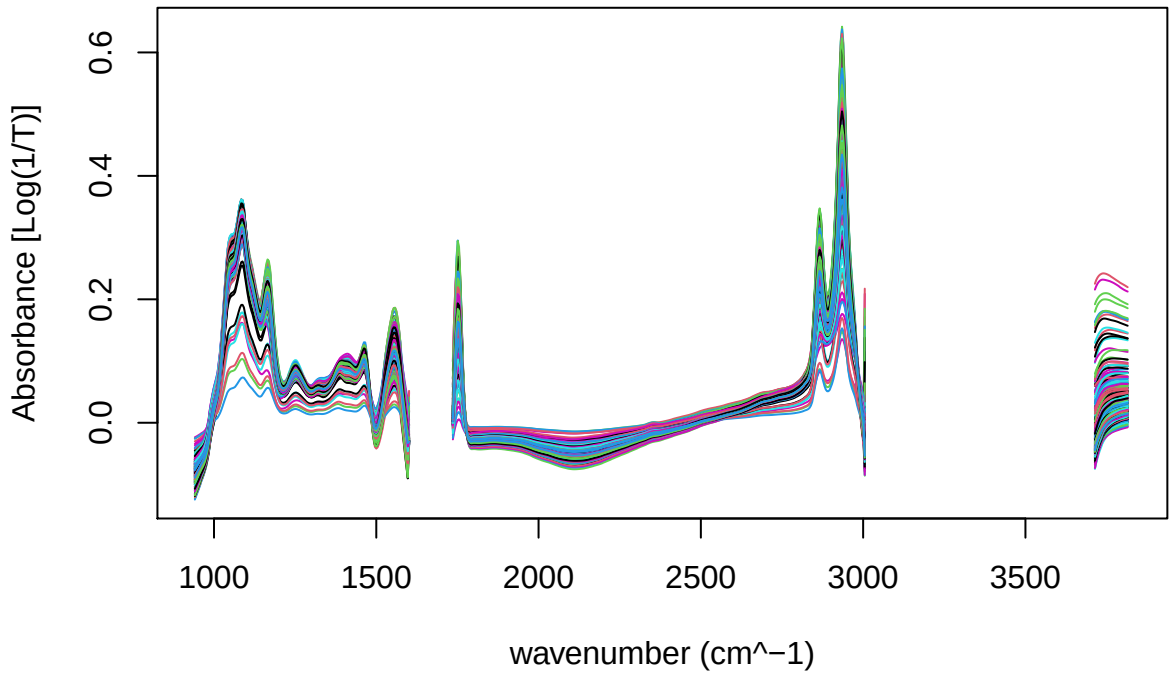
```
# Create data frame with wavelength data only
milk_waves <- na.omit(milkdata[, -c(1:18)])

# create vector of wavenumbers (drop "X" from columm names)
wavenums <- as.numeric(substring(names(milk_waves), 2))
# create vectors of spectral ranges (column indices)
range1 <- 1:172; range2 <- 173:502; range3 <- 503:529;
```

Before looking at clustering, we can visualise the spectra data using *matplot*. From the first plot below there is significant differences visible in the variation of the data across the spectrum (e.g. far more variation in absorbance values in 3714cm^{-1} - 3815cm^{-1} range than in the range of 2000cm^{-1} - 2500cm^{-1}).

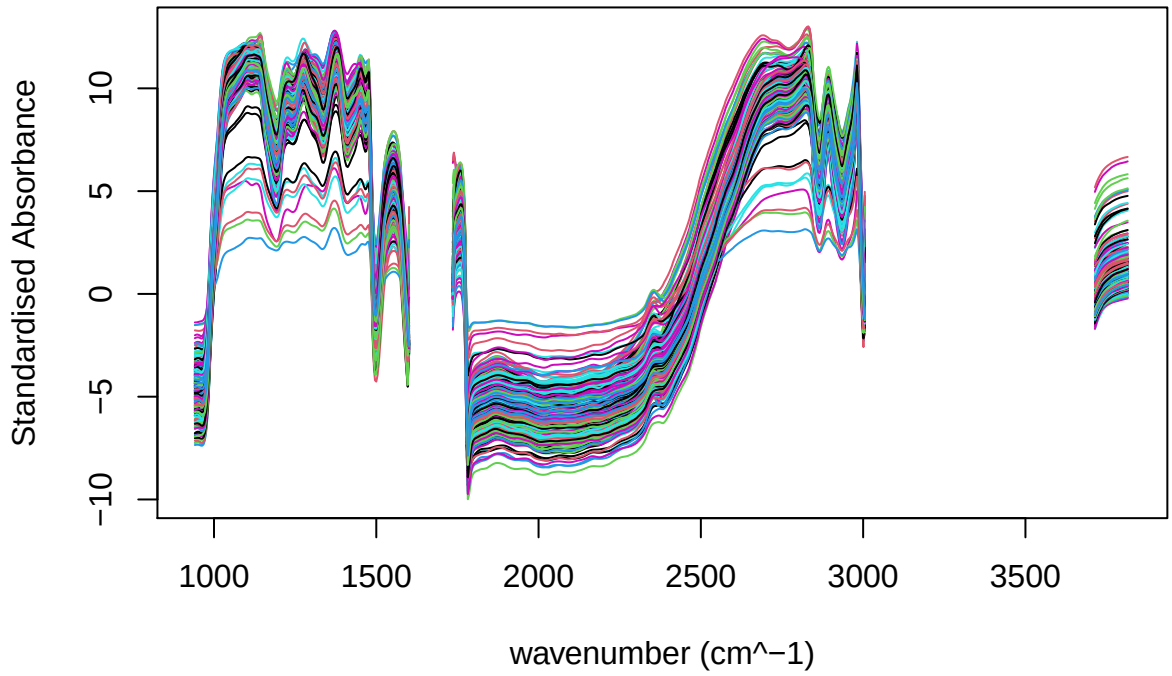
As such, it can be useful to standardise the data so that each feature (i.e. wavenumber) has an equal variance (equal to 1). We can do this by getting the standard deviation of each wavenumber column, and dividing the values in each column by the corresponding standard deviation. This can be helpful both in terms of visualisation and also when fitting the data to a model, as it gives equal weighting to each feature rather. The second plot below shows the MIR spectra data after it has been standardised.

While the mean values of the data at each wavenumber are preserved in the standardised data, it can be very helpful for visualisation purposes to also center the data so that the data at each wavenumber has a mean value of 0. The third plot below shows the MIR spectra data after it has been both standardised and centered. As all the data are on the same scale and are centered on the same mean value, it is easier to visualise the spread of the data across the full spectrum of wavenumbers.

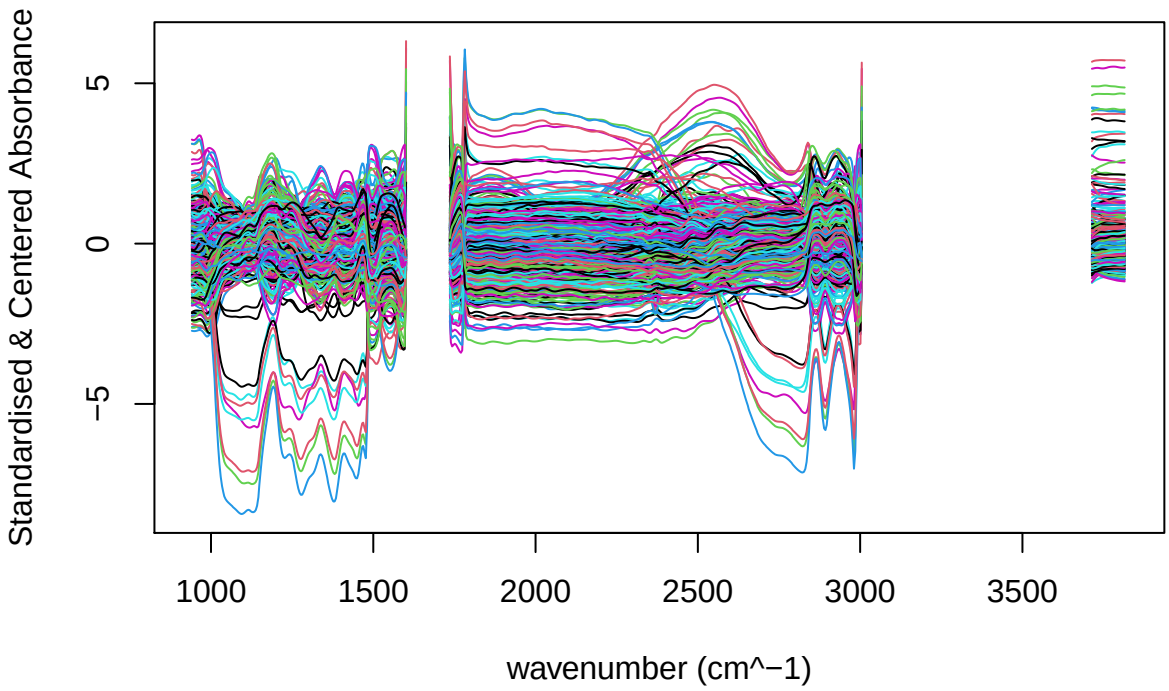
```
# plot wave data
matplot(wavenums, t(milk_waves), xlab = "wavenumber (cm^-1)", ylab = "Absorbance [Log(1/T)]", type="n")
matlines(wavenums[range1], t(milk_waves[,range1]), lwd = 1, type="l", lty = 1)
matlines(wavenums[range2], t(milk_waves[,range2]), lwd = 1, type="l", lty = 1)
matlines(wavenums[range3], t(milk_waves[,range3]), lwd = 1, type="l", lty = 1)
```




```
# Get standard deviation for each column
stDevWaves <- apply(milk_waves, 2, sd)
# divide each column of data frame by its standard deviation
stdwaves <- sweep(milk_waves, 2, stDevWaves, "/")

# plot standardised waves data
matplot(wavenums, t(stdwaves), xlab = "wavenumber (cm^-1)", ylab = "Standardised Absorbance", type="n")
matlines(wavenums[range1], t(stdwaves[,range1]), lwd = 1, type="l", lty = 1)
matlines(wavenums[range2], t(stdwaves[,range2]), lwd = 1, type="l", lty = 1)
matlines(wavenums[range3], t(stdwaves[,range3]), lwd = 1, type="l", lty = 1)
```



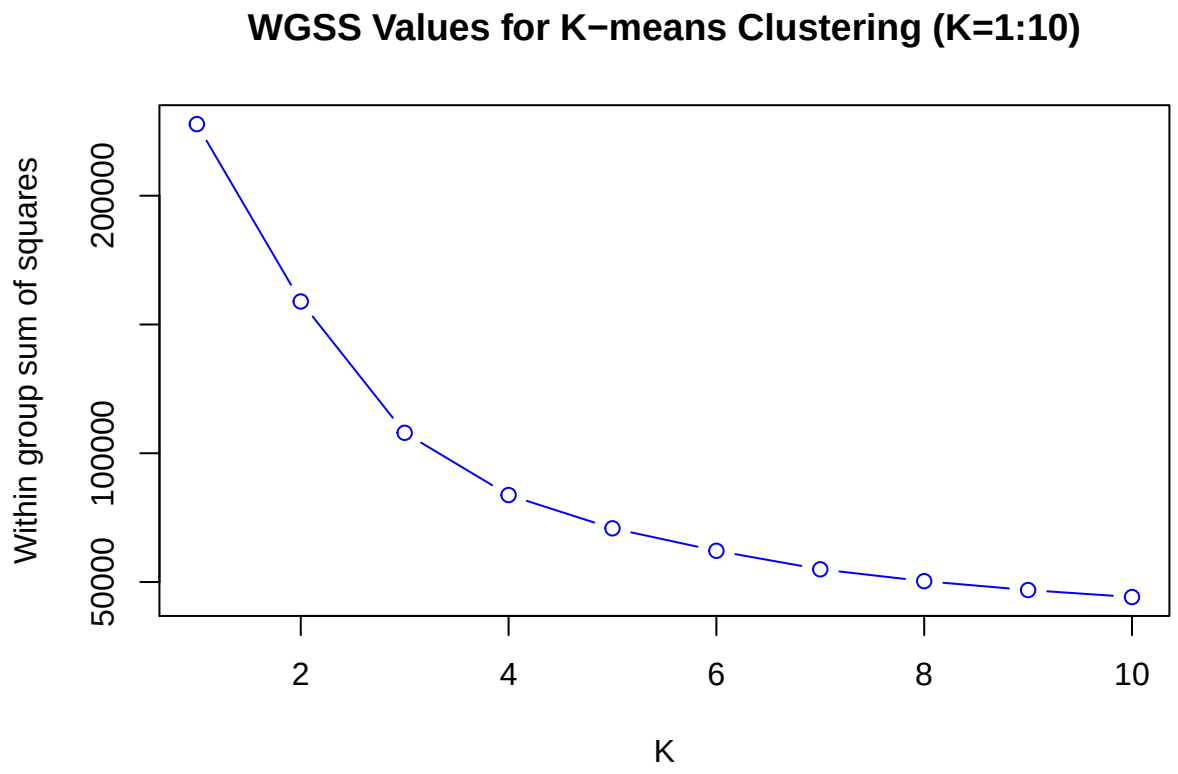
```
# plot standardised & centered waves data
matplot(wavenums, t(scale(stdwaves)), xlab = "wavenumber (cm^-1)", ylab = "Standardised & Centered Abso
matlines(wavenums[range1], t(scale(stdwaves[,range1])), lwd = 1, type="l", lty = 1)
matlines(wavenums[range2], t(scale(stdwaves[,range2])), lwd = 1, type="l", lty = 1)
matlines(wavenums[range3], t(scale(stdwaves[,range3])), lwd = 1, type="l", lty = 1)
```



K-means clustering is a partitioning clustering method that aims to separate data in to a specified number of clusters of similar observations. We need to define a suitable number of clusters for the K-means algorithm without assuming any knowledge of a suitable number of groups or clusters. One way to determine a suitable value is to calculate the within group sum of squares, which measures how tightly packed each of the clusters are.

We can repeat the K-means algorithm for a range of k values (number of clusters) and calculate the within group sum of squares value in each case. The plot below shows the within group sum of squares values for a range of k values of 1 to 10. We want to find the 'elbow' point in the plot, the k value where the rate of decrease of the within group sum of squares starts to reduce significantly. This appears to be at a value of k=3 below, or possibly k=4.

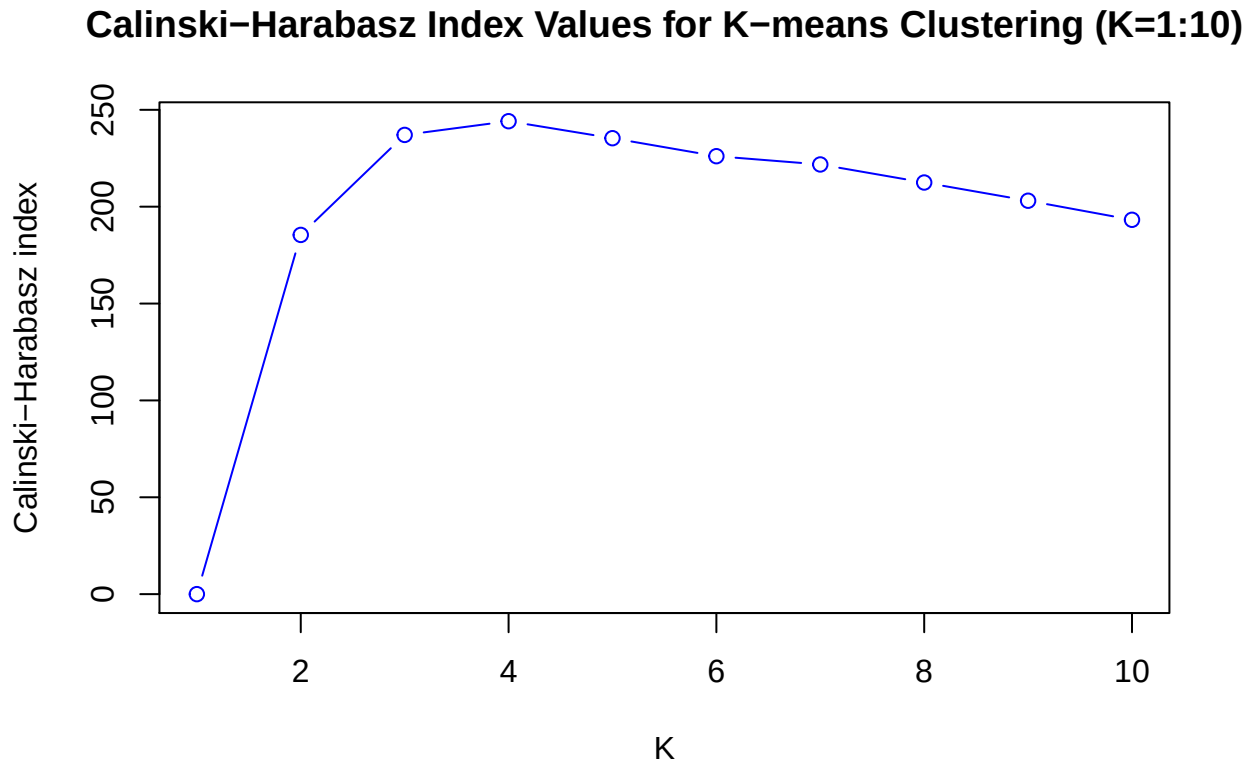
```
# create empty vector to store within group sum of squares values
WGSS <- rep(0,10)

# calculate WGSS for solution with 1 cluster
n <- nrow(stdwaves)
WGSS[1] <- (n-1) * sum(apply(stdwaves, 2, var))

# calculate WGSS values for 2 to 10 clusters
for(k in 2:10) {
  WGSS[k] <- sum(kmeans(stdwaves, centers = k, nstart = 50)$withinss)
}

# Plot WGSS vs K
plot(1:10, WGSS, type="b", xlab="K", ylab="Within group sum of squares",
     main = "WGSS Values for K-means Clustering (K=1:10)", col = "blue")
```



Another method that can be used to determine a suitable number of clusters is to look at the Calinski-Harabasz index. This is the ratio between the within group sum of squares and between sum of squares after they have each been scaled by their degrees of freedom.

A larger Calinski-Harabasz index value typically corresponds to a greater choice of number of clusters. The plot below shows the Calinski-Harabasz index value obtained for a range of k values of 1 to 10. Again, a value of k=3 or k=4 appear to be a reasonable number of clusters for this data set.

```
K <- 10
# create empty vectors to store within group sum of squares/between sum of squares
WSS <- BSS <- rep(NA, K)
for(k in 1:K) {
  # run kmeans for each value of k
  fit <- kmeans(stdwaves, centers = k, nstart = 50)
  WSS[k] <- fit$tot.withinss # store total within sum of squares
  BSS[k] <- fit$betweenss # store between sum of squares
}

# compute Calinski-Harabasz index for each value of K
N <- nrow(stdwaves)
ch <- (BSS / (1:K-1)) / (WSS / (N-1:K))
# the value of CH index for K = 1 is set to zero
ch[1] <- 0

par(mfrow=c(1,1))
# plot Calinski-Harabasz index for each value of K
plot(1:K, ch, type ="b", ylab ="Calinski-Harabasz index", xlab ="K", col = "blue",
     main = "Calinski-Harabasz Index Values for K-means Clustering (K=1:10)")
```



To determine whether the obtained clusters relate to any of the categorical covariates in the data set (*Breed*, *Date_of_sampling*, *Parity*, *Milking_Time*), we can compare clusters obtained using the K-means algorithm for k=3 and k=4 as suggested from the plots above, with each of the categorical covariates.

To measure the similarity between the clusters and the categories, the adjusted Rand index can be calculated.

The Rand index is a measure of the agreement between 2 data groupings, while the adjusted Rand index is a measure of this agreement after correcting for agreement by chance.

From the table below, the strongest agreement was identified between the clustering of k=3 and the *Milking_Time* categorical covariate, with an adjusted Rand index value of 0.25. The remaining categorical covariates returned adjusted Rand index values close to 0, indicating that they were not similar to the clustering structures obtained using K-means.

```
# try kmeans clustering with k=3:4
fit3 <- kmeans(stdwaves, centers = 3, nstart = 50)
fit4 <- kmeans(stdwaves, centers = 4, nstart = 50)

# create empty data frame and vector for storing adjusted rand values
randTable <- data.frame()
new_row <- NULL

# loop through the 4 categorical traits in the data set
for(i in 1:4){
  # get adjusted rand index values for K=3:4 for current variable
  new_row <- c(
          classAgreement(table(fit3$cluster, milkdata[,i]))$crand,
          classAgreement(table(fit4$cluster, milkdata[,i]))$crand)

  # get adjusted rand index values for K=3:4 for current variable
  randTable <- rbind(randTable, new_row)
}

# round values in table to 2 decimal places
randTable <- round(randTable, 2)
# set row and column names
rownames(randTable) <- names(milkdata[,1:4])
names(randTable) <- c("K = 3", "K = 4")
# print table of adjusted rand index values
kable(randTable, align = 'c')
```

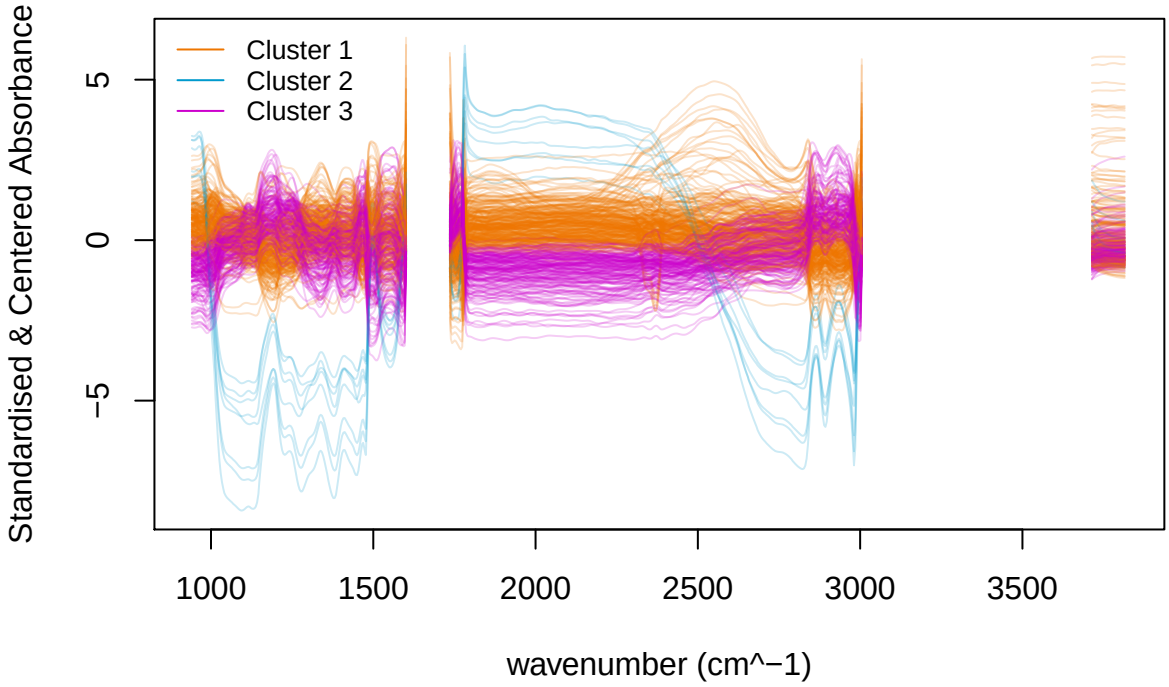
	K = 3	K = 4
Breed	0.06	0.04
Date_of_sampling	0.03	0.05
Parity	0.02	0.03
Milking_Time	0.25	0.20

We can use *matplot* to visualise the spectral data again, but this time we will colour each observation's curve based on its associated cluster, as shown in the forst plot below. The grouping appears to separate the spectra relatively well.

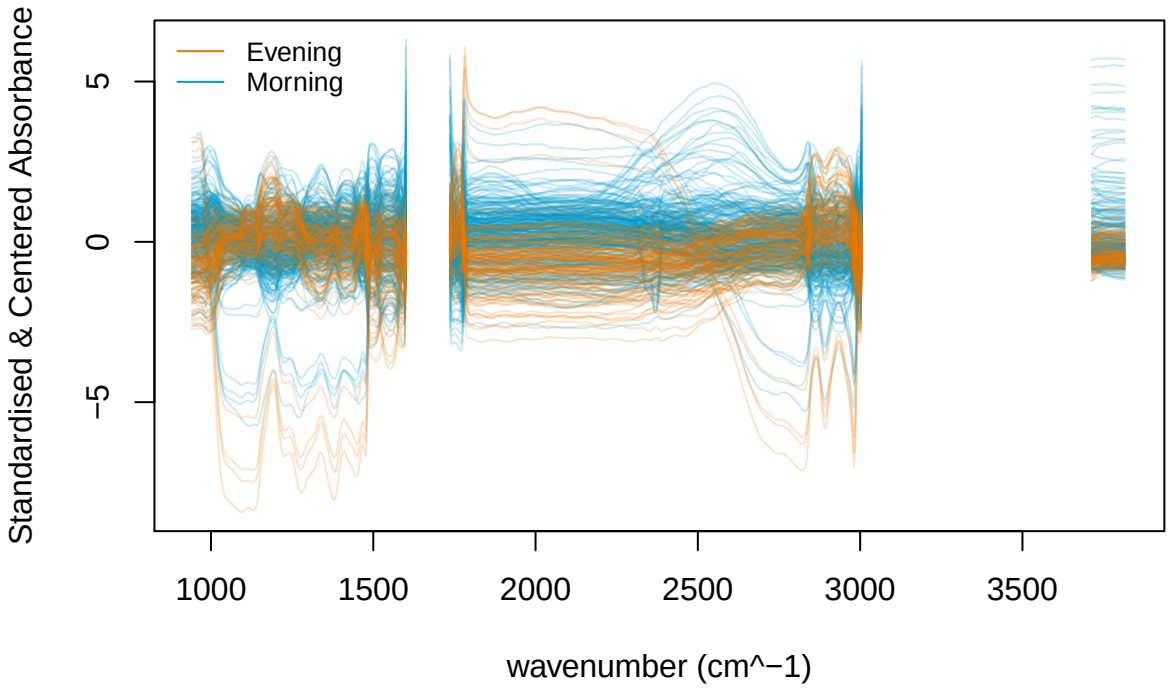
Repeating this but colouring each curve based on Milking Time and comparing to the first plot, it appears that the "Evening" values typically correspond to one of the clustered, and "Morning" values typically correspond to another cluster.

We can view what frequency each of the Evening and Morning observations overlap with each of the clusters using the *table* function, as shown below. While the clusters don't separate the morning and evening samples perfectly, one of the clusters does capture the majority of the Evening observations and another cluster captures the majority of the Morning values.

```
# check if any NA values are in milking time column, save index values
MT_NA <- which(is.na(milkdata$Milking_Time))

# symbols and colours vector
symb <-c(15, 16, 17, 18)
col <-c("darkorange2","deepskyblue3","magenta3", "darkgreen")

#  create factor of fit3 cluster data
fit3clust <- as.factor((fit3$cluster)[-MT_NA])
levels(fit3clust) <- c("Cluster 1", "Cluster 2", "Cluster 3")

# plot standardised spectra with colours corresponding to clusters (K = 3)
matplot(wavenums, t(scale(stdwaves[-MT_NA,])), type="n", xlab = "wavenumber (cm^-1)", ylab = "Standardis
legend("topleft", legend = c("Cluster 1", "Cluster 2", "Cluster 3"), col = col[1:3], lty = 1, bty = "n"
matlines(wavenums[range1], t(scale(stdwaves[-MT_NA,range1])), col = adjustcolor(col[fit3clust], 0.2), lw
matlines(wavenums[range2], t(scale(stdwaves[-MT_NA,range2])), col = adjustcolor(col[fit3clust], 0.2), lw
matlines(wavenums[range3], t(scale(stdwaves[-MT_NA,range3])), col = adjustcolor(col[fit3clust], 0.2), lw
```




```
# plot standardised spectra with colours corresponding to Milking_Time group
matplot(wavenums, t(scale(stdwaves)), type = "n", xlab = "wavenumber (cm^-1)", ylab = "Standardised & C
legend("topleft", legend = c("Evening", "Morning"), col = col[1:2], lty = 1, bty = "n", cex = 0.8)
matlines(wavenums[range1], t(scale(stdwaves[-MT_NA,range1])), col = adjustcolor(col[(milkdata$Milking_T
matlines(wavenums[range2], t(scale(stdwaves[-MT_NA,range2])), col = adjustcolor(col[(milkdata$Milking_T
matlines(wavenums[range3], t(scale(stdwaves[-MT_NA,range3])), col = adjustcolor(col[(milkdata$Milking_T
```




```
# compare kmeans clustering (k=3) with Milking_Time
tab <- table(fit3clust, (milkdata$Milking_Time)[-MT_NA])

# print table
kable(tab, align = 'c')
```

	Evening	Morning
Cluster 1	35	222
Cluster 2	4	4
Cluster 3	97	67

```
# print rand index and adjusted rand index values
cat("Rand index:", classAgreement(tab)$rand)
```

```
## Rand index: 0.6270614
```

```
cat("Adjusted Rand Index", classAgreement(tab)$crand)
```

```
## Adjusted Rand Index 0.2532924
```

3. Classification

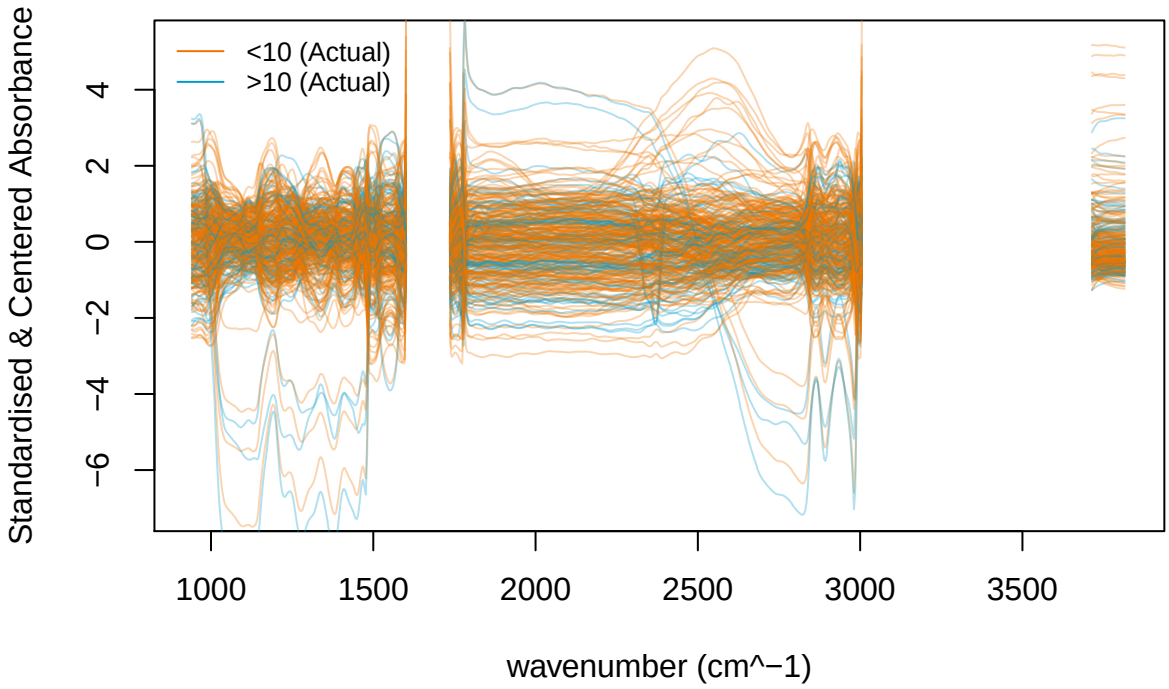
Next we will try to classify milk samples as having a heat stability of less than 10 minutes. As the data set gives a heat stability time for each observation (in minutes), we need to create a vector of values corresponding to whether an observation's heat stability is less than or greater than 10 minutes, assigning a value of 1 for samples with heat stability less than 10 minutes, and 2 otherwise.

```
# Create vector of ones
HeatStability10 <- rep(1, nrow(stdwaves))
# set value to 2 if corresponding row has Heat Stability > 10
# value is 1 if <=10, value is 2 if >10
HeatStability10[milkdata$Heat_stability > 10] <- 2

# create new data frame which is a copy of stdwaves
stdwaves1 <- stdwaves
# add HeatStability10 column to the data frame
stdwaves1$HS10 <- HeatStability10
```

As the data is highly collinear, I will first attempt to use K nearest neighbours classification on the spectral data. We need to specify the number of neighbours, k, to be considered for the algorithm, but this is not always easy to choose.

Before fitting any models, I will first split the data into a test and a training data set. The training data will be used for training and validating the model. After the best model has been fitted, a final test of the model will be performed using the test data.

```
# proportion of rows to be reserved for test data
test_split <- 0.25

# number of rows to be reserved for test data
num_test_inds <- floor(test_split * nrow(stdwaves1))

# generate indices for training and test data
test_ind <- sample(1:nrow(stdwaves1), num_test_inds)
train_ind <- setdiff(1:nrow(stdwaves1), test_ind)

# plot standardised spectra with colours corresponding to Heat Stability less than / greater than 10 mi
matplot(wavenums, t(scale(stdwaves[test_ind,])), type="n", xlab = "wavenumber (cm^-1)", ylab = "Standar
legend("topleft", legend = c("<10 (Actual)", ">10 (Actual)"), col = col[1:2], lty = 1, bty = "n", cex =
matlines(wavenums[range1], t(scale(stdwaves[train_ind, range1])), col = adjustcolor(col[HeatStability10
matlines(wavenums[range2], t(scale(stdwaves[train_ind, range2])), col = adjustcolor(col[HeatStability10
matlines(wavenums[range3], t(scale(stdwaves[train_ind, range3])), col = adjustcolor(col[HeatStability10
```

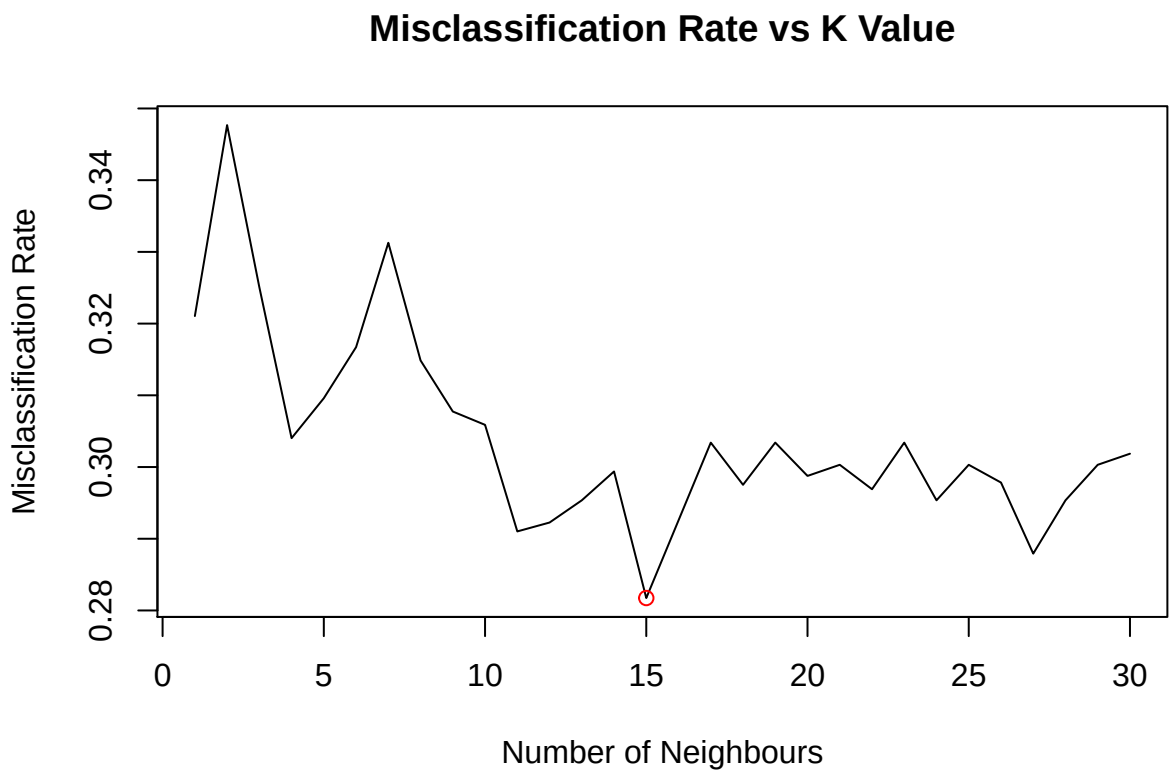


To determine a suitable k value, we can test a range of different k-values with the k nearest neighbours algorithm and calculate the misclassification rate in each case. The *knn.cv* function can be used to perform leave-one-out cross-validatory classification on the training data for each value.

As results can vary with every iteration, I will perform the algorithm 10 times on each k-value and take the mean misclassification value.

```
# empty vector to hold misclassification values for each k
vec <- NULL

# test values of neighbours considered for knn from 1 to 30
for(kk in 1:30) {

  # empty vector to hold misclassification values
  vec2 <- NULL

  # run knn algorithm 10 times for each k, take average misclassification
  for(i in 1:10){
    # Perform k nearest neighbours cross validation (leave one out)
    knn.res <- knn.cv(stdwaves[train_ind,], cl=HeatStability10[train_ind], k=kk)
    # create table comparing knn result to actual heat stability classification vector
    tab <- table(knn.res, HeatStability10[train_ind])
    # add misclassification rate of current knn classification
    vec2 <- c(vec2, 1 - sum(diag(tab)) / sum(tab) )

  }
  # add average misclassification rate of current k to vec
  vec <- c(vec, mean(vec2))
}

# plot misclassification rate versus K
plot(1:30, vec, main = "Misclassification Rate vs K Value", xlab = "Number of Neighbours", ylab = "Misc
# plot red point at min. value
```

```
points(which(vec == min(vec))[1], min(vec), col = "red")
```



The plot shows minimum misclassification rate at K=27. We can test this again on the training data, and create a confusion matrix comparing knn classified results to the actual heat stability classes.

```
# perform knn cross validation again with optimum k (=27)
knn.res <- knn.cv(stdwaves[train_ind,], cl=HeatStability10[train_ind], k=27)

# create table of confusion matrix
tab <- table(knn.res, HeatStability10[train_ind])

# rename table rows and columns
rownames(tab) <- c("LT10 (KNN)", "GT10 (KNN)")
colnames(tab) <- c("LT10 (Actual)", "GT10 (Actual)")

# print table
kable(tab, align = 'c')
```

	LT10 (Actual)	GT10 (Actual)
LT10 (KNN)	214	87
GT10 (KNN)	6	16

```
cat("Misclassification Rate (KNN): ", 1 - sum(diag(tab)) / sum(tab))
```

```
## Misclassification Rate (KNN):  0.2879257
```

We can also test knn on our test data, using K=27 as above.

```
# use knn to predict test data classes from training data, K=27
knn.res <- knn(stdwaves[train_ind,], stdwaves[test_ind,], cl = HeatStability10[train_ind], k = 27)

# create confusion matrix table
tab <- table(knn.res, HeatStability10[test_ind])

# rename table rows and columns
rownames(tab) <- c("LT10 (KNN)", "GT10 (KNN)")
colnames(tab) <- c("LT10 (Actual)", "GT10 (Actual)")

# print table
kable(tab, align = 'c')
```

	LT10 (Actual)	GT10 (Actual)
LT10 (KNN)	62	34
GT10 (KNN)	4	7

```
cat("Test Misclassification Rate (KNN): ", 1 - sum(diag(tab)) / sum(tab))
```

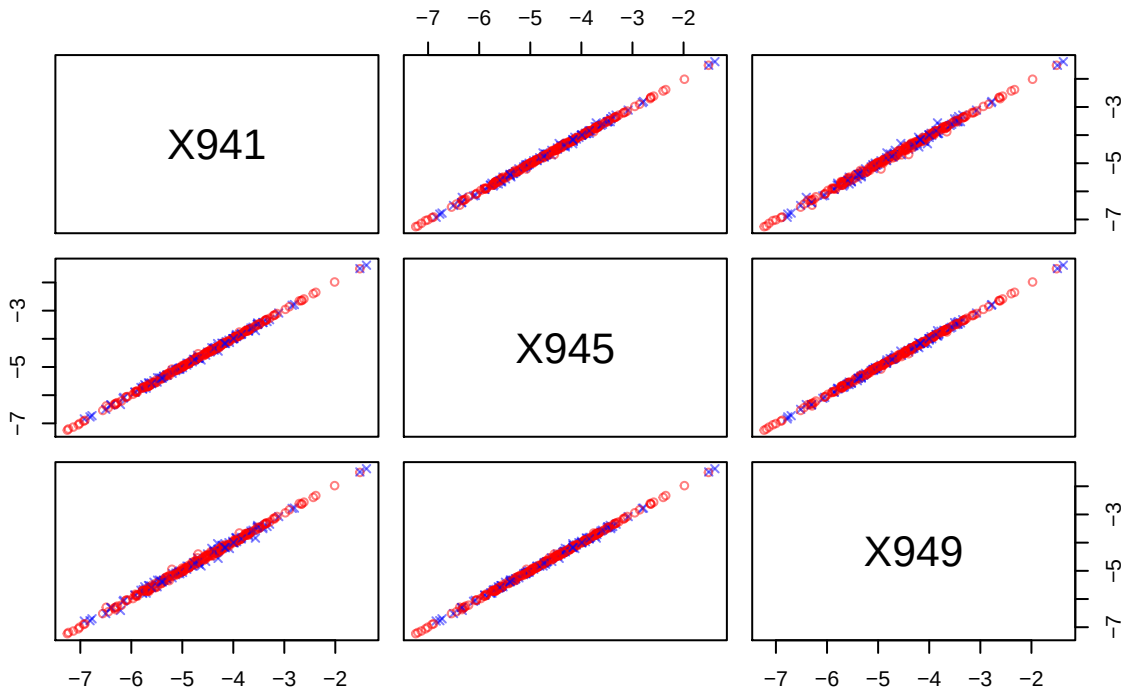
```
## Test Misclassification Rate (KNN):  0.3551402
```

The misclassification rate of the heat stability data using knn is still relatively high.

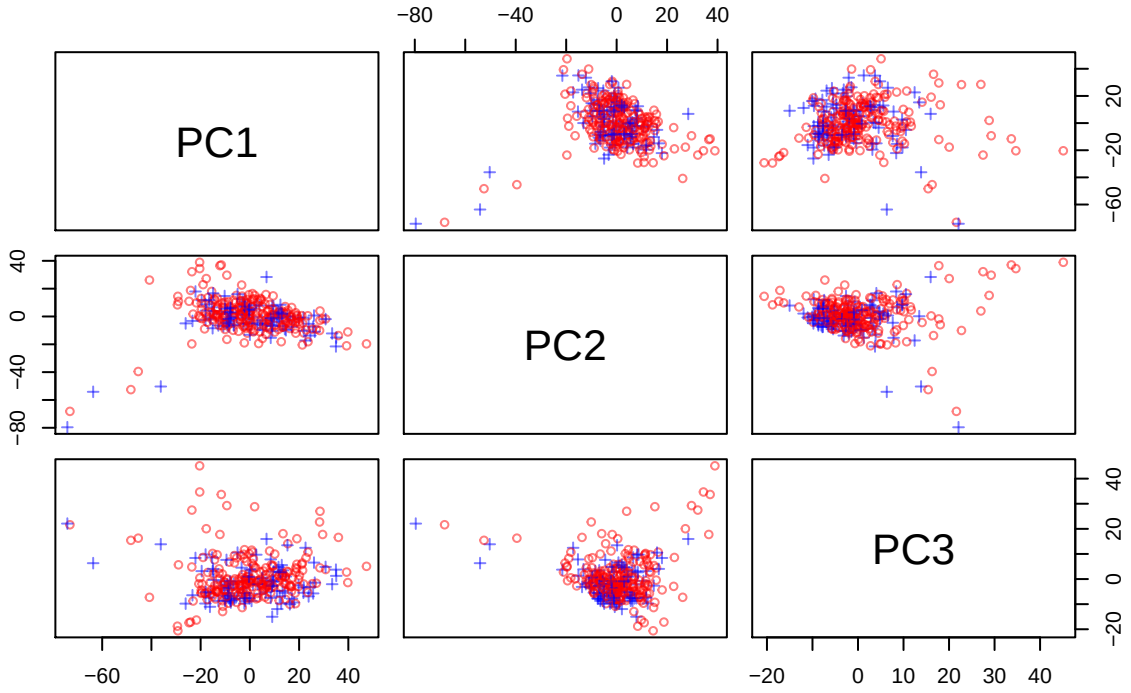
We can try other approaches such as Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA), however one issue with using those methods on the current data is that the spectral data is highly collinear, and there are a large number of columns.

To deal with this, we can first use Principal Component Analysis (PCA) to represent the data in the spectral columns with a smaller number of independent columns of data. The pairs plots below show the first 4 columns of the spectral data, and the first 4 principal components.

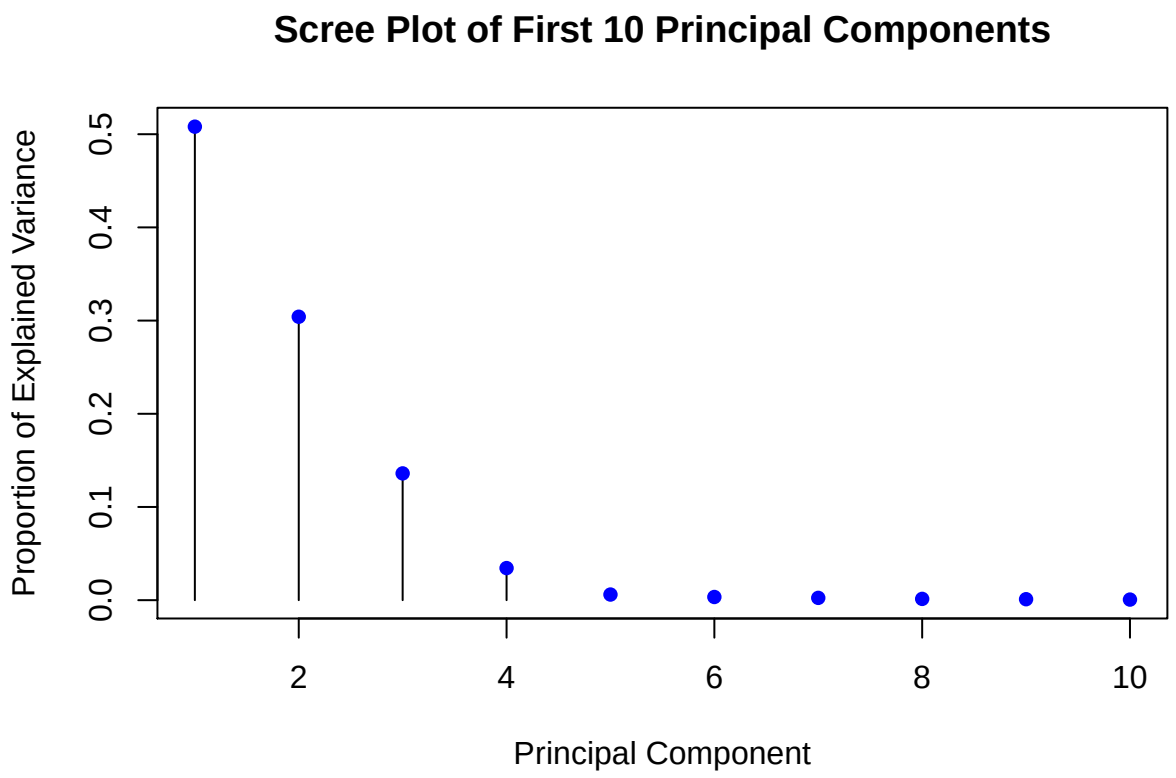
```
# perform PCA on training data
fit <- prcomp(stdwaves1[train_ind,])

# get principal components for training data
newspectra = predict(fit)

# pairs plot of first 4 columns of spectral data
pairs(stdwaves[train_ind,1:3], pch=c(1,4)[HeatStability10[train_ind]],
      col=adjustcolor(c("red", "blue"), 0.5)[HeatStability10[train_ind]], cex = 0.8,
      main = "Pairs Plot of First 3 Wavenumbers")
```




```
# pairs plot of first 4 Principal Components
pairs(newspectra[,1:3], pch=c(1,3)[HeatStability10[train_ind]],
      col=adjustcolor(c("red", "blue"), 0.5)[HeatStability10[train_ind]], cex = 0.8,
      main = "Pairs Plot of First 4 Principal Components")
```

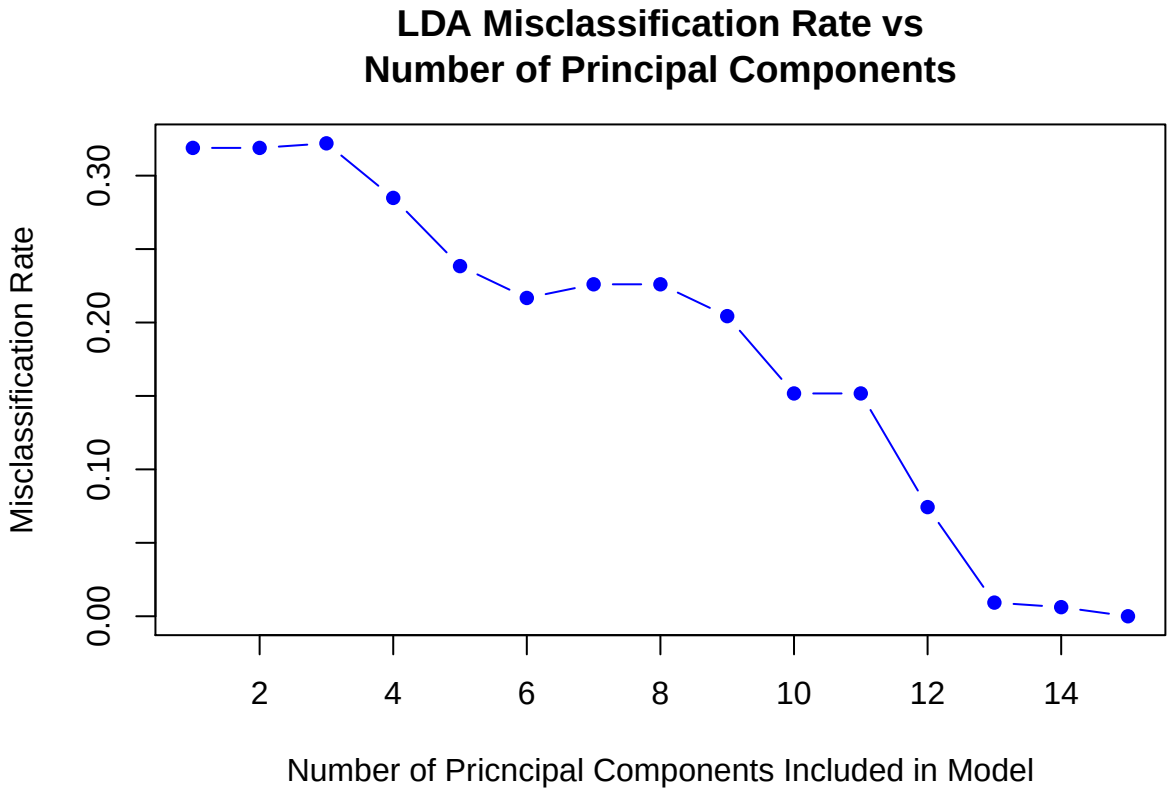


```r
# calculate proportion of variation explained by first 10 PCs
propvar <- rep(0,10)
for(i in 1:10) propvar[i] <- (((fit$sdev)[i])^2)/(sum(fit$sdev^2))

# Plot the proportion of variance explained by first 10 PCs
plot(1:10, propvar, type = "h", main = "Scree Plot of First 10 Principal Components",
     xlab = "Principal Component", ylab = "Proportion of Explained Variance")
points(1:10, propvar, col = "blue", pch = 16)
```



We can use linear discriminant analysis on a subset of the principal components, but to determine a suitable number of components to use we can fit multiple models and calculate and plot the misclassification rate in each case. Performing LDA using leave-one-out cross validation on the training data, a misclassification rate value of 0 was achieved after 13 principal components were used when training the model.

```r
#LDA PCA
vec <- NULL
# Loop through 1:15 (number of PCs to be included in model)
for(i in 1:15) {
  # linear discriminant analysis with leave one out cross validation
  lda.res.cv <- lda((stdwaves1$HS10)[train_ind] ~ newspectra[,1:i], CV = TRUE)
  # cross tabulation between lda classification and actual heat stability classes
  tab <- table(lda.res.cv$class, HeatStability10[train_ind])
  # add misclassification to vector
  vec <- c(vec, 1 - sum(diag(tab)) / sum(tab))
}

# plot misclassification rate vs number of
plot(1:15, vec, type = "b", main = "LDA Misclassification Rate vs\nNumber of Principal Components",
     xlab = "Number of Pricncipal Components Included in Model", ylab = "Misclassification Rate", pch =
```



We can now test our model obtained from cross-validation on the training data. First, we must transform the test data based on the principal component analysis carried out on the training data.

Then we can fit the model to the training data, using a subset of the principal components. Below I have chosen to use the first 10. After fitting the model, we can obtain predictions for the test data, and compare the results to the actual heat stability classes of the test data. A confusion matrix of the results is shown below, along with the calculated misclassification rate.

```
# get principal components for test data
testspectra = predict(fit, newdata = stdwaves1[test_ind,])

# Use optimum number of PCs in LDA
lda.fit <- lda(newspectra[,1:10], (stdwaves1$HS10)[train_ind])

# predict classes for test data
preds <- predict(lda.fit, testspectra[,1:10])$class

# cross tabulation between lda classification and actual heat stability classes
tab <- table(preds, HeatStability10[test_ind])
# rename table rows and columns
rownames(tab) <- c("LT10 (LDA)", "GT10 (LDA)")
colnames(tab) <- c("LT10 (Actual)", "GT10 (Actual)")

# plot confusion matrix
kable(tab, align = 'c')
```

	LT10 (Actual)	GT10 (Actual)
LT10 (LDA)	59	16
GT10 (LDA)	7	25

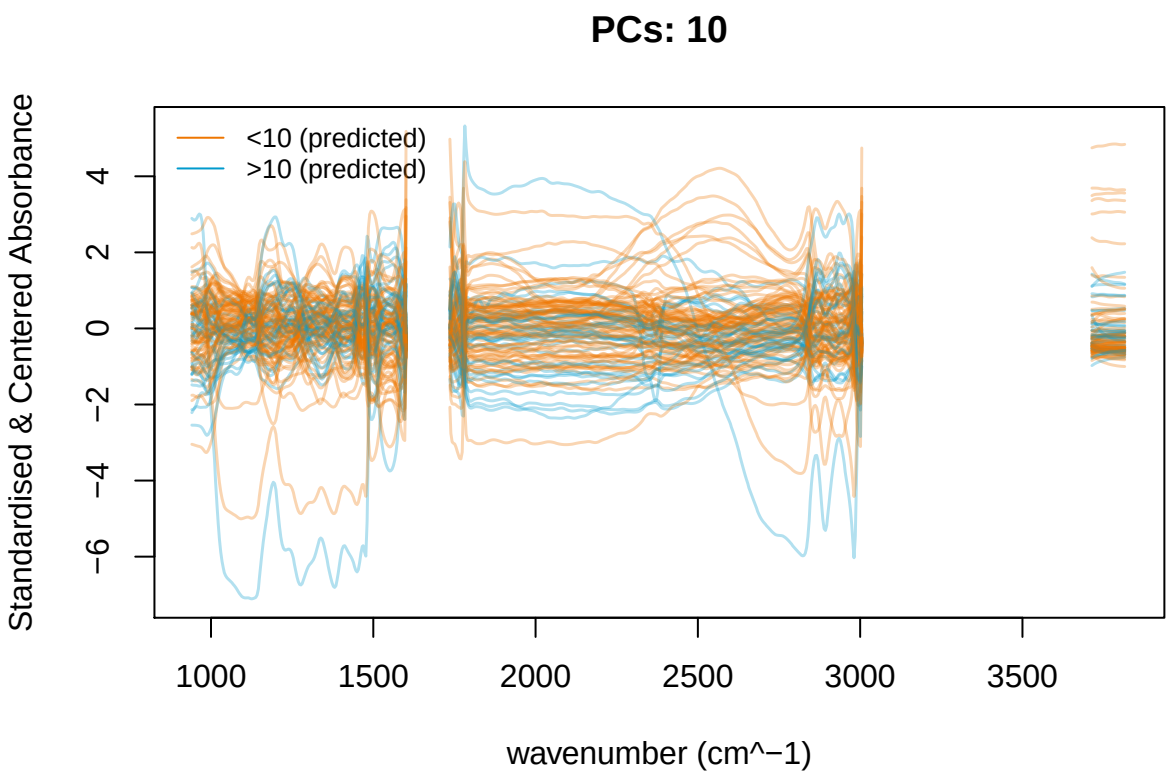
```
# print misclassification rate for test data
cat("LDA PDA Test", 1 - sum(diag(tab)) / sum(tab))
```

```
## LDA PDA Test 0.2149533
```

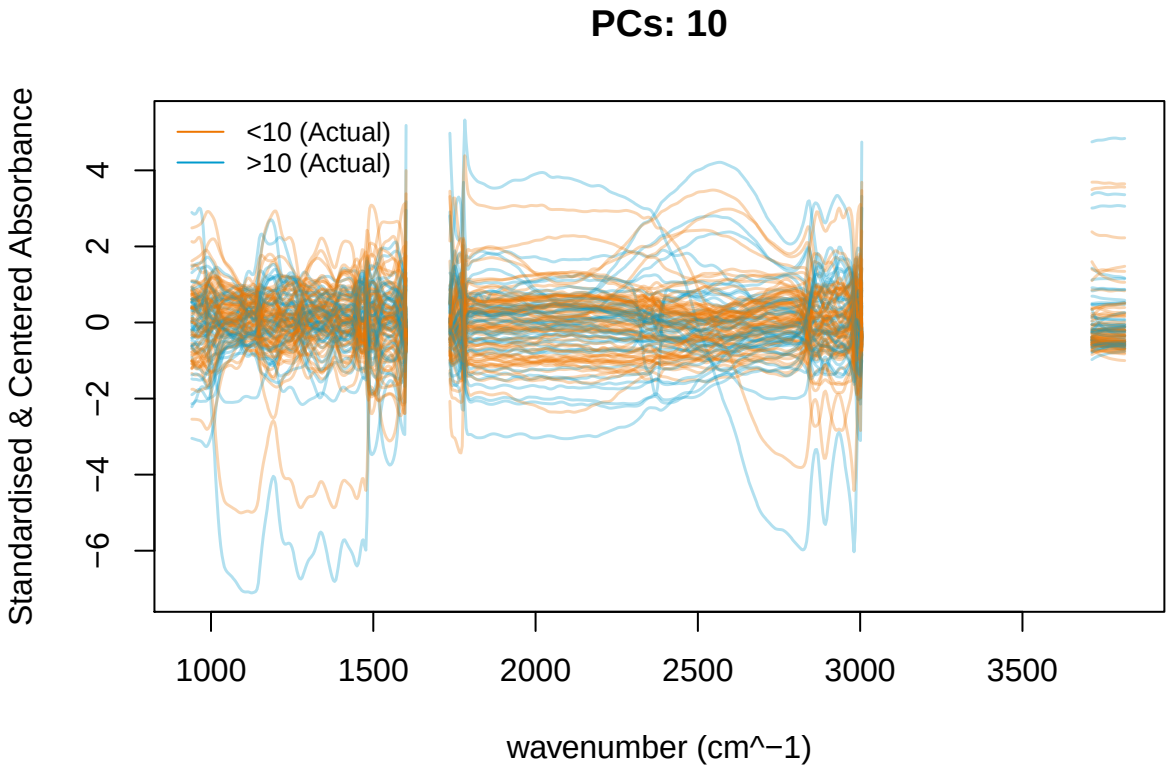
We can improve the LDA classification by using more principal components when fitting the model. Firstly, I have plotted the MIR spectra of the samples below, first with the colour for each observation based on the predicted heat stability class, and then for the actual heat stability class, when using 10 principal components during the fitting of the LDA classification model. Then I plotted the spectra again, but with colour based on whether the observation was correctly identified or not.

I then refitted the model twice, first using 11 principal components from the training data and then using 12. In both cases, I used the models to predict the heat stability class of each of the observations in the test data set, and I again produced spectral plots with observations colours based on whether they were classified correctly or incorrectly.

```
# plot standardised spectra with colours corresponding to predicted heat stability classes
matplot(wavenums, t(scale(stdwaves[test_ind,])), type="n", xlab = "wavenumber (cm^-1)", ylab = "Standard
legend("topleft", legend = c("<10 (predicted)", ">10 (predicted)"), col = col[1:2], lty = 1, bty = "n",
matlines(wavenums[range1], t(scale(stdwaves[test_ind, range1])), col = adjustcolor(col[preds], 0.3), lw
matlines(wavenums[range2], t(scale(stdwaves[test_ind, range2])), col = adjustcolor(col[preds], 0.3), lw
matlines(wavenums[range3], t(scale(stdwaves[test_ind, range3])), col = adjustcolor(col[preds], 0.3), lw
```



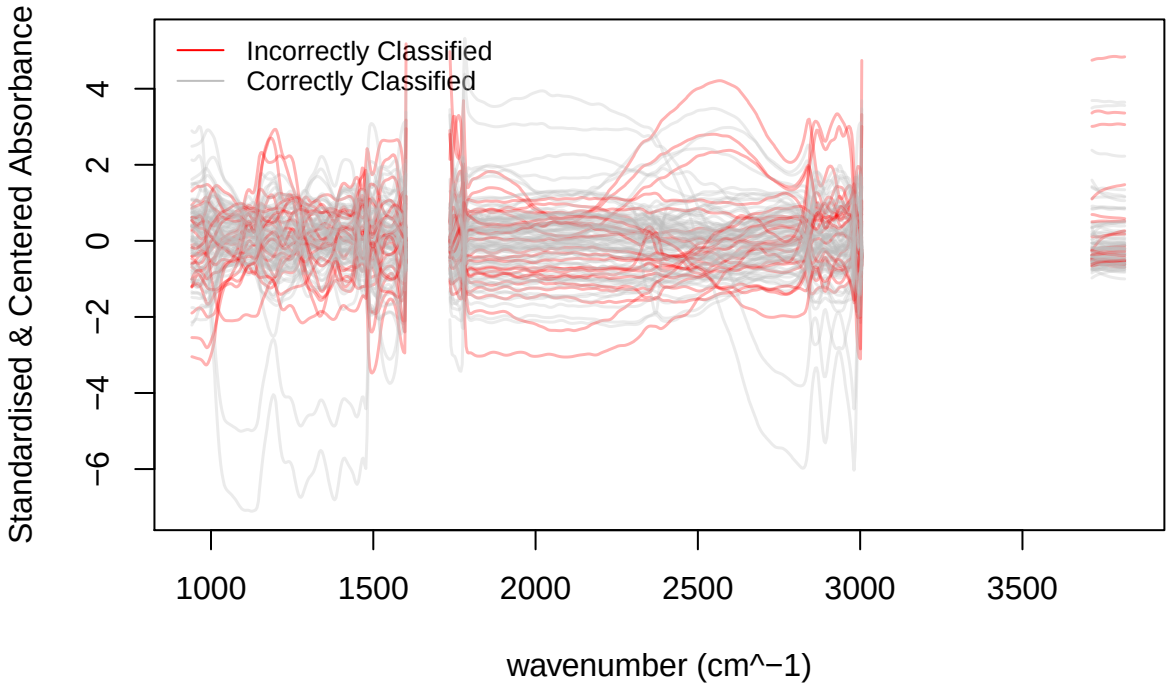
```
# plot standardised spectra with colours corresponding to actual heat stability classes
matplot(wavenums, t(scale(stdwaves[test_ind,])), type="n", xlab = "wavenumber (cm^-1)", ylab = "Standard
legend("topleft", legend = c("<10 (Actual)", ">10 (Actual)"), col = col[1:2], lty = 1, bty = "n", cex =
matlines(wavenums[range1], t(scale(stdwaves[test_ind, range1])), col = adjustcolor(col[HeatStability10[
matlines(wavenums[range2], t(scale(stdwaves[test_ind, range2])), col = adjustcolor(col[HeatStability10[
matlines(wavenums[range3], t(scale(stdwaves[test_ind, range3])), col = adjustcolor(col[HeatStability10[
```



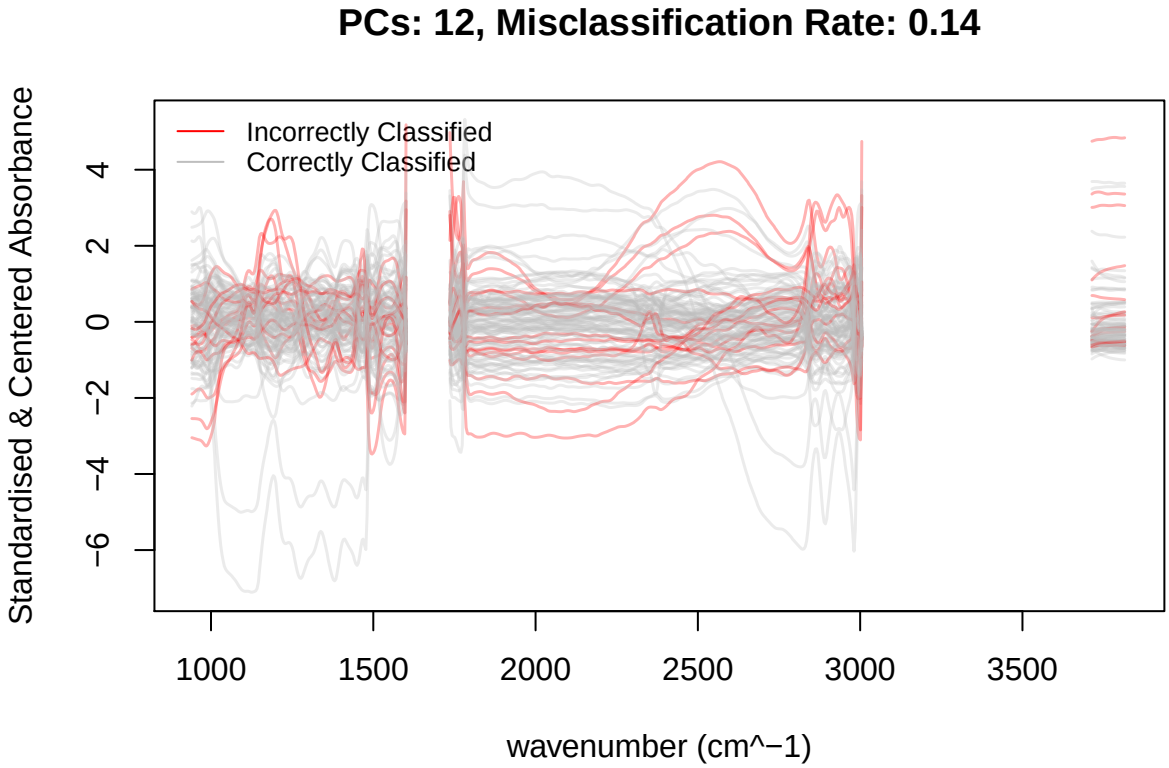

```
tab <- table(preds, HeatStability10[test_ind])
tab
```

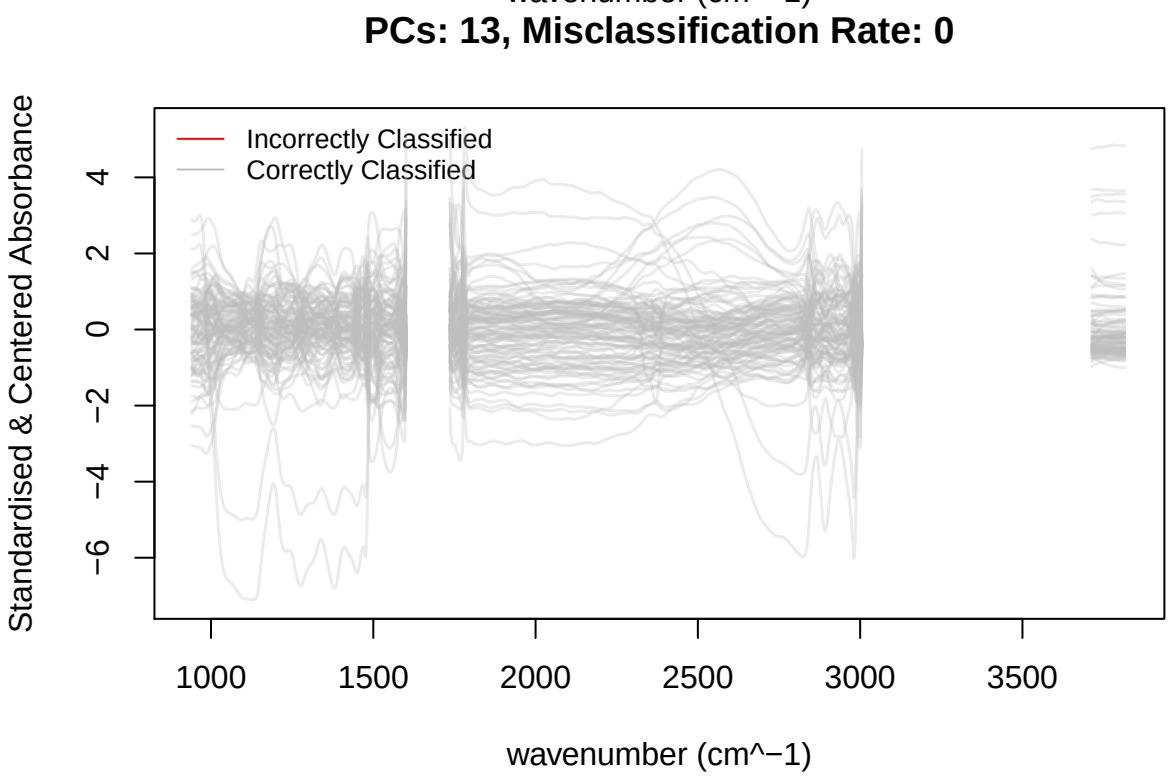
```
##
## preds  1  2
##     1 59 16
##     2  7 25
```

```
# plot standardised spectra with colours corresponding to correct/incorrect predictions
matplot(wavenums, t(scale(stdwaves[test_ind,])), type="n", xlab = "wavenumber (cm^-1)", ylab = "Standard
legend("topleft", legend = c("Incorrectly Classified", "Correctly Classified"), col = c("red", "gray"),
matlines(wavenums[range1], t(scale(stdwaves[test_ind, range1])), col = adjustcolor(c("red", "gray")[(pr
matlines(wavenums[range2], t(scale(stdwaves[test_ind, range2])), col = adjustcolor(c("red", "gray")[(pr
matlines(wavenums[range3], t(scale(stdwaves[test_ind, range3])), col = adjustcolor(c("red", "gray")[(pr
```




```
# Test using 11 and 12 PCs in LDA
for(i in 12:13){
  lda.fit <- lda(newspectra[,1:i], (stdwaves1$HS10)[train_ind])
  # predict classes for test data
  preds <- predict(lda.fit, testspectra[,1:i])$class
  tab <- table(preds, HeatStability10[test_ind])

  # plot standardised spectra with colours corresponding to correct/incorrect predictions
  matplot(wavenums, t(scale(stdwaves[test_ind,])), type="n", xlab = "wavenumber (cm^-1)", ylab = "Standa
          main = paste0("PCs: ", i,", Misclassification Rate: ", round(1 - sum(diag(tab)) / sum(tab), 2)
  legend("topleft", legend = c("Incorrectly Classified", "Correctly Classified"), col = c("red", "gray"
  matlines(wavenums[range1], t(scale(stdwaves[test_ind, range1])),
           col = adjustcolor(c("red", "gray")[(preds == HeatStability10[test_ind])+1], 0.3), lwd = 1.5,
  matlines(wavenums[range2], t(scale(stdwaves[test_ind, range2])),
           col = adjustcolor(c("red", "gray")[(preds == HeatStability10[test_ind])+1], 0.3), lwd = 1.5,
  matlines(wavenums[range3], t(scale(stdwaves[test_ind, range3])),
           col = adjustcolor(c("red", "gray")[(preds == HeatStability10[test_ind])+1], 0.3), lwd = 1.5,
}
```





From the results above, LDA appears to be very effective at classifying the heat stability of milk samples based on their associated MIR spectrum, particularly when using at least the first 13 principal components from the spectral data after performing principal component analysis.